Framework

AWS Well-Architected Framework



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Well-Architected Framework: Framework

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Abstract and introduction 1
Introduction 1
Definitions 2
On architecture 4
General design principles 6
The pillars of the framework 8
Operational excellence 8
Design principles 9
Definition 10
Best practices 11
Resources 20
Security 20
Design principles 21
Definition 21
Best practices 22
Resources 31
Reliability 32
Design principles 32
Definition 33
Best practices 34
Resources 39
Performance efficiency 39
Design principles 40
Definition 40
Best practices 41
Resources 46
Cost optimization 46
Design principles 47
Definition 48
Best practices 48
Resources 54
Sustainability 55
Design principles 55
Definition 56

Best practices 57
Resources 63
The review process 64
Conclusion 66
Contributors 67
Further reading 68
Document revisions 69
Appendix: Questions and best practices 73
Operational excellence 73
Organization 73
Prepare 130
Operate 198
Evolve 241
Security 260
Security foundations 261
Identity and access management 285
Detection 343
Infrastructure protection 357
Data protection 383
Incident response 417
Application security 440
Reliability 464
Foundations 464
Workload architecture 504
Change management 553
Failure management 600
Performance efficiency 701
Architecture selection 702
Compute and hardware 716
Data management 734
Networking and content delivery 757
Process and culture 786
Cost optimization 803
Practice Cloud Financial Management 803
Expenditure and usage awareness 826
Cost-effective resources 868

Manage demand and supply resources 909
Optimize over time 921
Sustainability 929
Region selection 930
Alignment to demand 932
Software and architecture 946
Data 958
Hardware and services 976
Process and culture 986
Notices 997
AWS Glossary 998

AWS Well-Architected Framework

Publication date: **November 6, 2024** (*Document revisions*)

The AWS Well-Architected Framework helps you understand the pros and cons of decisions you make while building systems on AWS. By using the Framework you will learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems in the cloud.

Introduction

The AWS Well-Architected Framework helps you understand the pros and cons of decisions you make while building systems on AWS. Using the Framework helps you learn architectural best practices for designing and operating secure, reliable, efficient, cost-effective, and sustainable workloads in the AWS Cloud. It provides a way for you to consistently measure your architectures against best practices and identify areas for improvement. The process for reviewing an architecture is a constructive conversation about architectural decisions, and is not an audit mechanism. We believe that having well-architected systems greatly increases the likelihood of business success.

AWS Solutions Architects have years of experience architecting solutions across a wide variety of business verticals and use cases. We have helped design and review thousands of customers' architectures on AWS. From this experience, we have identified best practices and core strategies for architecting systems in the cloud.

The AWS Well-Architected Framework documents a set of foundational questions that help you to understand if a specific architecture aligns well with cloud best practices. The framework provides a consistent approach to evaluating systems against the qualities you expect from modern cloud-based systems, and the remediation that would be required to achieve those qualities. As AWS continues to evolve, and we continue to learn more from working with our customers, we will continue to refine the definition of well-architected.

This framework is intended for those in technology roles, such as chief technology officers (CTOs), architects, developers, and operations team members. It describes AWS best practices and strategies to use when designing and operating a cloud workload, and provides links to further implementation details and architectural patterns. For more information, see the AWS Well-Architected homepage.

AWS also provides a service for reviewing your workloads at no charge. The AWS Well-Architected Tool (AWS WA Tool) is a service in the cloud that provides a consistent process for you to review and measure your architecture using the AWS Well-Architected Framework. The AWS WA Tool provides recommendations for making your workloads more reliable, secure, efficient, and cost-effective.

To help you apply best practices, we have created AWS Well-Architected Labs, which provides you with a repository of code and documentation to give you hands-on experience implementing best practices. We also have teamed up with select AWS Partner Network (APN) Partners, who are members of the AWS Well-Architected Partner program. These AWS Partners have deep AWS knowledge, and can help you review and improve your workloads.

Definitions

Every day, experts at AWS assist customers in architecting systems to take advantage of best practices in the cloud. We work with you on making architectural trade-offs as your designs evolve. As you deploy these systems into live environments, we learn how well these systems perform and the consequences of those trade-offs.

Based on what we have learned, we have created the AWS Well-Architected Framework, which provides a consistent set of best practices for customers and partners to evaluate architectures, and provides a set of questions you can use to evaluate how well an architecture is aligned to AWS best practices.

The AWS Well-Architected Framework is based on six pillars — operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability.

Table 1. The pillars of the AWS Well-Architected Framework

Name	Description
Operational excellence	The ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value.
Security	The security pillar describes how to take advantage of cloud technologies to protect

Name	Description
	data, systems, and assets in a way that can improve your security posture.
Reliability	The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle . This paper provides in-depth, best practice guidance for implementing reliable workloads on AWS.
Performance efficiency	The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
Cost optimization	The ability to run systems to deliver business value at the lowest price point.
Sustainability	The ability to continually improve sustainab ility impacts by reducing energy consumption and increasing efficiency across all component s of a workload by maximizing the benefits from the provisioned resources and minimizin g the total resources required.

In the AWS Well-Architected Framework, we use these terms:

- A **component** is the code, configuration, and AWS Resources that together deliver against a requirement. A component is often the unit of technical ownership, and is decoupled from other components.
- The term **workload** is used to identify a set of components that together deliver business value. A workload is usually the level of detail that business and technology leaders communicate about.

- We think about **architecture** as being how components work together in a workload. How components communicate and interact is often the focus of architecture diagrams.
- **Milestones** mark key changes in your architecture as it evolves throughout the product lifecycle (design, implementation, testing, go live, and in production).
- Within an organization the **technology portfolio** is the collection of workloads that are required for the business to operate.
- The **level of effort** is categorizing the amount of time, effort, and complexity a task requires for implementation. Each organization needs to consider the size and expertise of the team and the complexity of the workload for additional context to properly categorize the level of effort for the organization.
  - **High:** The work might take multiple weeks or multiple months. This could be broken out into multiple stories, releases, and tasks.
  - **Medium:** The work might take multiple days or multiple weeks. This could be broken out into multiple releases and tasks.
  - **Low:** The work might take multiple hours or multiple days. This could be broken out into multiple tasks.

When architecting workloads, you make trade-offs between pillars based on your business context. These business decisions can drive your engineering priorities. You might optimize to improve sustainability impact and reduce cost at the expense of reliability in development environments, or, for mission-critical solutions, you might optimize reliability with increased costs and sustainability impact. In ecommerce solutions, performance can affect revenue and customer propensity to buy. Security and operational excellence are generally not traded-off against the other pillars.

On architecture

In on-premises environments, customers often have a central team for technology architecture that acts as an overlay to other product or feature teams to verify they are following best practice. Technology architecture teams typically include a set of roles such as: Technical Architect (infrastructure), Solutions Architect (software), Data Architect, Networking Architect, and Security Architect. Often these teams use TOGAF or the Zachman Framework as part of an enterprise architecture capability.

At AWS, we prefer to distribute capabilities into teams rather than having a centralized team with that capability. There are risks when you choose to distribute decision making authority, for

example, verifying that teams are meeting internal standards. We mitigate these risks in two ways. First, we have *practices* (ways of doing things, process, standards, and accepted norms) that focus on allowing each team to have that capability, and we put in place experts who verify that teams raise the bar on the standards they need to meet. Second, we implement *mechanisms* that carry out automated checks to verify standards are being met.

> "Good intentions never work, you need good mechanisms to make anything happen" — Jeff Bezos.

This means replacing a human's best efforts with mechanisms (often automated) that check for compliance with rules or process. This distributed approach is supported by the Amazon leadership principles, and establishes a culture across all roles that *works back* from the customer. Working backward is a fundamental part of our innovation process. We start with the customer and what they want, and let that define and guide our efforts. Customer-obsessed teams build products in response to a customer need.

For architecture, this means that we expect every team to have the capability to create architectures and to follow best practices. To help new teams gain these capabilities or existing teams to raise their bar, we activate access to a virtual community of principal engineers who can review their designs and help them understand what AWS best practices are. The principal engineering community works to make best practices visible and accessible. One way they do this, for example, is through lunchtime talks that focus on applying best practices to real examples. These talks are recorded and can be used as part of onboarding materials for new team members.

AWS best practices emerge from our experience running thousands of systems at internet scale. We prefer to use data to define best practice, but we also use subject matter experts, like principal engineers, to set them. As principal engineers see new best practices emerge, they work as a community to verify that teams follow them. In time, these best practices are formalized into our internal review processes, and also into mechanisms that enforce compliance. The Well-Architected Framework is the customer-facing implementation of our internal review process, where we have codified our principal engineering thinking across field roles, like Solutions Architecture and internal engineering teams. The Well-Architected Framework is a scalable mechanism that lets you take advantage of these learnings.

By following the approach of a principal engineering community with distributed ownership of architecture, we believe that a Well-Architected enterprise architecture can emerge that is driven

by customer need. Technology leaders (such as a CTOs or development managers), carrying out Well-Architected reviews across all your workloads will permit you to better understand the risks in your technology portfolio. Using this approach, you can identify themes across teams that your organization could address by mechanisms, training, or lunchtime talks where your principal engineers can share their thinking on specific areas with multiple teams.

General design principles

The Well-Architected Framework identifies a set of general design principles to facilitate good design in the cloud:

- **Stop guessing your capacity needs**: If you make a poor capacity decision when deploying a workload, you might end up sitting on expensive idle resources or dealing with the performance implications of limited capacity. With cloud computing, these problems can go away. You can use as much or as little capacity as you need, and scale in and out automatically.
- **Test systems at production scale**: In the cloud, you can create a production-scale test environment on demand, complete your testing, and then decommission the resources. Because you only pay for the test environment when it's running, you can simulate your live environment for a fraction of the cost of testing on premises.
- **Automate with architectural experimentation in mind**: Automation permits you to create and replicate your workloads at low cost and avoid the expense of manual effort. You can track changes to your automation, audit the impact, and revert to previous parameters when necessary.
- **Consider evolutionary architectures**: In a traditional environment, architectural decisions are often implemented as static, onetime events, with a few major versions of a system during its lifetime. As a business and its context continue to evolve, these initial decisions might hinder the system's ability to deliver changing business requirements. In the cloud, the capability to automate and test on demand lowers the risk of impact from design changes. This permits systems to evolve over time so that businesses can take advantage of innovations as a standard practice.
- **Drive architectures using data**: In the cloud, you can collect data on how your architectural choices affect the behavior of your workload. This lets you make fact-based decisions on how to improve your workload. Your cloud infrastructure is code, so you can use that data to inform your architecture choices and improvements over time.
- **Improve through game days**: Test how your architecture and processes perform by regularly scheduling game days to simulate events in production. This will help you understand where

improvements can be made and can help develop organizational experience in dealing with events.

The pillars of the framework

Creating a software system is a lot like constructing a building. If the foundation is not solid, structural problems can undermine the integrity and function of the building. When architecting technology solutions, if you neglect the six pillars of operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability, it can become challenging to build a system that delivers on your expectations and requirements. Incorporating these pillars into your architecture will help you produce stable and efficient systems. This will allow you to focus on the other aspects of design, such as functional requirements.

Pillars

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Sustainability

Operational excellence

Operational excellence (OE) is a commitment to build software correctly while consistently delivering a great customer experience. The operational excellence pillar contains best practices for organizing your team, designing your workload, operating it at scale, and evolving it over time.

The operational excellence pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Operational Excellence Pillar whitepaper.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

The following are design principles for operational excellence in the cloud:

- **Organize teams around business outcomes:** The ability of a team to achieve business outcomes comes from leadership vision, effective operations, and a business-aligned operating model. Leadership should be fully invested and committed to a CloudOps transformation with a suitable cloud operating model that incentivizes teams to operate in the most efficient way and meet business outcomes. The right operating model uses people, process, and technology capabilities to scale, optimize for productivity, and differentiate through agility, responsiveness, and adaptation. The organization's long-term vision is translated into goals that are communicated across the enterprise to stakeholders and consumers of your cloud services. Goals and operational KPIs are aligned at all levels. This practice sustains the long-term value derived from implementing the following design principles.
- **Implement observability for actionable insights:** Gain a comprehensive understanding of workload behavior, performance, reliability, cost, and health. Establish key performance indicators (KPIs) and leverage observability telemetry to make informed decisions and take prompt action when business outcomes are at risk. Proactively improve performance, reliability, and cost based on actionable observability data.
- **Safely automate where possible:** In the cloud, you can apply the same engineering discipline that you use for application code to your entire environment. You can define your entire workload and its operations (applications, infrastructure, configuration, and procedures) as code, and update it. You can then automate your workload’s operations by initiating them in response to events. In the cloud, you can employ automation safety by configuring guardrails, including rate control, error thresholds, and approvals. Through effective automation, you can achieve consistent responses to events, limit human error, and reduce operator toil.
- **Make frequent, small, reversible changes:** Design workloads that are scalable and loosely coupled to permit components to be updated regularly. Automated deployment techniques together with smaller, incremental changes reduces the blast radius and allows for faster reversal when failures occur. This increases confidence to deliver beneficial changes to your workload while maintaining quality and adapting quickly to changes in market conditions.
- **Refine operations procedures frequently:** As you evolve your workloads, evolve your operations appropriately. As you use operations procedures, look for opportunities to improve them. Hold regular reviews and validate that all procedures are effective and that teams are familiar with them. Where gaps are identified, update procedures accordingly. Communicate procedural

updates to all stakeholders and teams. Gamify your operations to share best practices and educate teams.

- **Anticipate failure:** Maximize operational success by driving failure scenarios to understand the workload's risk profile and its impact on your business outcomes. Test the effectiveness of your procedures and your team's response against these simulated failures. Make informed decisions to manage open risks that are identified by your testing.
- **Learn from all operational events and metrics:** Drive improvement through lessons learned from all operational events and failures. Share what is learned across teams and through the entire organization. Learnings should highlight data and anecdotes on how operations contribute to business outcomes.
- **Use managed services:** Reduce operational burden by using AWS managed services where possible. Build operational procedures around interactions with those services.

Definition

There are four best practice areas for operational excellence in the cloud:

- **Organization**
- **Prepare**
- **Operate**
- **Evolve**

Your organization's leadership defines business objectives. Your organization must understand requirements and priorities and use these to organize and conduct work to support the achievement of business outcomes. Your workload must emit the information necessary to support it. Implementing services to achieve integration, deployment, and delivery of your workload will create an increased flow of beneficial changes into production by automating repetitive processes.

There may be risks inherent in the operation of your workload. Understand those risks and make an informed decision to enter production. Your teams must be able to support your workload. Business and operational metrics derived from desired business outcomes will permit you to understand the health of your workload, your operations activities, and respond to incidents. Your priorities will change as your business needs and business environment changes. Use these as a feedback loop to continually drive improvement for your organization and the operation of your workload.

Best practices

> **Note**
>
> All operational excellence questions have the OPS prefix as a shorthand for the pillar.

Topics

- Organization
- Prepare
- Operate
- Evolve

Organization

Your teams must have a shared understanding of your entire workload, their role in it, and shared business goals to set the priorities that will achieve business success. Well-defined priorities will maximize the benefits of your efforts. Evaluate internal and external customer needs involving key stakeholders, including business, development, and operations teams, to determine where to focus efforts. Evaluating customer needs will verify that you have a thorough understanding of the support that is required to achieve business outcomes. Verify that you are aware of guidelines or obligations defined by your organizational governance and external factors, such as regulatory compliance requirements and industry standards that may mandate or emphasize specific focus. Validate that you have mechanisms to identify changes to internal governance and external compliance requirements. If no requirements are identified, validate that you have applied due diligence to this determination. Review your priorities regularly so that they can be updated as needs change.

Evaluate threats to the business (for example, business risk and liabilities, and information security threats) and maintain this information in a risk registry. Evaluate the impact of risks, and tradeoffs between competing interests or alternative approaches. For example, accelerating speed to market for new features may be emphasized over cost optimization, or you may choose a relational database for non-relational data to simplify the effort to migrate a system without refactoring. Manage benefits and risks to make informed decisions when determining where to focus efforts. Some risks or choices may be acceptable for a time, it may be possible to mitigate associated risks,

or it may become unacceptable to permit a risk to remain, in which case you will take action to address the risk.

Your teams must understand their part in achieving business outcomes. Teams must understand their roles in the success of other teams, the role of other teams in their success, and have shared goals. Understanding responsibility, ownership, how decisions are made, and who has authority to make decisions will help focus efforts and maximize the benefits from your teams. The needs of a team will be shaped by the customer they support, their organization, the makeup of the team, and the characteristics of their workload. It's unreasonable to expect a single operating model to be able to support all teams and their workloads in your organization.

Verify that there are identified owners for each application, workload, platform, and infrastructure component, and that each process and procedure has an identified owner responsible for its definition, and owners responsible for their performance.

Having understanding of the business value of each component, process, and procedure, of why those resources are in place or activities are performed, and why that ownership exists will inform the actions of your team members. Clearly define the responsibilities of team members so that they may act appropriately and have mechanisms to identify responsibility and ownership. Have mechanisms to request additions, changes, and exceptions so that you do not constrain innovation. Define agreements between teams describing how they work together to support each other and your business outcomes.

Provide support for your team members so that they can be more effective in taking action and supporting your business outcomes. Engaged senior leadership should set expectations and measure success. Senior leadership should be the sponsor, advocate, and driver for the adoption of best practices and evolution of the organization. Let team members take action when outcomes are at risk to minimize impact and encourage them to escalate to decision makers and stakeholders when they believe there is a risk so that it can be addressed and incidents avoided. Provide timely, clear, and actionable communications of known risks and planned events so that team members can take timely and appropriate action.

Encourage experimentation to accelerate learning and keep team members interested and engaged. Teams must grow their skill sets to adopt new technologies, and to support changes in demand and responsibilities. Support and encourage this by providing dedicated structured time for learning. Verify that your team members have the resources, both tools and team members, to be successful and scale to support your business outcomes. Leverage cross-organizational diversity to seek multiple unique perspectives. Use this perspective to increase innovation, challenge your

assumptions, and reduce the risk of confirmation bias. Grow inclusion, diversity, and accessibility within your teams to gain beneficial perspectives.

If there are external regulatory or compliance requirements that apply to your organization, you should use the resources provided by AWS Cloud Compliance to help educate your teams so that they can determine the impact on your priorities. The Well-Architected Framework emphasizes learning, measuring, and improving. It provides a consistent approach for you to evaluate architectures, and implement designs that will scale over time. AWS provides the AWS Well-Architected Tool to help you review your approach before development, the state of your workloads before production, and the state of your workloads in production. You can compare workloads to the latest AWS architectural best practices, monitor their overall status, and gain insight into potential risks. AWS Trusted Advisor is a tool that provides access to a core set of checks that recommend optimizations that may help shape your priorities. Business and Enterprise Support customers receive access to additional checks focusing on security, reliability, performance, cost-optimization, and sustainability that can further help shape their priorities.

AWS can help you educate your teams about AWS and its services to increase their understanding of how their choices can have an impact on your workload. Use the resources provided by AWS Support (AWS Knowledge Center, AWS Discussion Forums, and AWS Support Center) and AWS Documentation to educate your teams. Reach out to AWS Support through AWS Support Center for help with your AWS questions. AWS also shares best practices and patterns that we have learned through the operation of AWS in The Amazon Builders' Library. A wide variety of other useful information is available through the AWS Blog and The Official AWS Podcast. AWS Training and Certification provides some training through self-paced digital courses on AWS fundamentals. You can also register for instructor-led training to further support the development of your teams' AWS skills.

Use tools or services that permit you to centrally govern your environments across accounts, such as AWS Organizations, to help manage your operating models. Services like AWS Control Tower expand this management capability by allowing you to define blueprints (supporting your operating models) for the setup of accounts, apply ongoing governance using AWS Organizations, and automate provisioning of new accounts. Managed Services providers such as AWS Managed Services, AWS Managed Services Partners, or Managed Services Providers in the AWS Partner Network, provide expertise implementing cloud environments, and support your security and compliance requirements and business goals. Adding Managed Services to your operating model can save you time and resources, and lets you keep your internal teams lean and focused on strategic outcomes that will differentiate your business, rather than developing new skills and capabilities.

The following questions focus on these considerations for operational excellence. (For a list of operational excellence questions and best practices, see the Appendix.)

OPS 1: How do you determine what your priorities are?

Everyone must understand their part in achieving business success. Have shared goals in order to set priorities for resources. This will maximize the benefits of your efforts.

OPS 2: How do you structure your organization to support your business outcomes?

Your teams must understand their part in achieving business outcomes. Teams must understand their roles in the success of other teams, the role of other teams in their success, and have shared goals. Understanding responsibility, ownership, how decisions are made, and who has authority to make decisions will help focus efforts and maximize the benefits from your teams.

OPS 3: How does your organizational culture support your business outcomes?

Provide support for your team members so that they can be more effective in taking action and supporting your business outcome.

You might find that you want to emphasize a small subset of your priorities at some point in time. Use a balanced approach over the long term to verify the development of needed capabilities and management of risk. Review your priorities regularly and update them as needs change. When responsibility and ownership are undefined or unknown, you are at risk of both not performing necessary action in a timely fashion and of redundant and potentially conflicting efforts emerging to address those needs. Organizational culture has a direct impact on team member job satisfaction and retention. Activate the engagement and capabilities of your team members to achieve the success of your business. Experimentation is required for innovation to happen and turn ideas into outcomes. Recognize that an undesired result is a successful experiment that has identified a path that will not lead to success.

Prepare

To prepare for operational excellence, you have to understand your workloads and their expected behaviors. You will then be able to design them to provide insight to their status and build the procedures to support them.

Design your workload so that it provides the information necessary for you to understand its internal state (for example, metrics, logs, events, and traces) across all components in support of observability and investigating issues. Observability goes beyond simple monitoring, providing a comprehensive understanding of a system's internal workings based on its external outputs. Rooted in metrics, logs, and traces, observability offers profound insights into system behavior and dynamics. With effective observability, teams can discern patterns, anomalies, and trends, allowing them to proactively address potential issues and maintain optimal system health. Identifying key performance indicators (KPIs) is pivotal to ensure alignment between monitoring activities and business objectives. This alignment ensures that teams are making data-driven decisions using metrics that genuinely matter, optimizing both system performance and business outcomes. Furthermore, observability empowers businesses to be proactive rather than reactive. Teams can understand the cause-and-effect relationships within their systems, predicting and preventing issues rather than just reacting to them. As workloads evolve, it's essential to revisit and refine the observability strategy, ensuring it remains relevant and effective.

Adopt approaches that improve the flow of changes into production and that achieves refactoring, fast feedback on quality, and bug fixing. These accelerate beneficial changes entering production, limit issues deployed, and activate rapid identification and remediation of issues introduced through deployment activities or discovered in your environments.

Adopt approaches that provide fast feedback on quality and achieves rapid recovery from changes that do not have desired outcomes. Using these practices mitigates the impact of issues introduced through the deployment of changes. Plan for unsuccessful changes so that you are able to respond faster if necessary and test and validate the changes you make. Be aware of planned activities in your environments so that you can manage the risk of changes impacting planned activities. Emphasize frequent, small, reversible changes to limit the scope of change. This results in faster troubleshooting and remediation with the option to roll back a change. It also means you are able to get the benefit of valuable changes more frequently.

Evaluate the operational readiness of your workload, processes, procedures, and personnel to understand the operational risks related to your workload. Use a consistent process (including manual or automated checklists) to know when you are ready to go live with your workload or

a change. This will also help you to find any areas that you must make plans to address. Have runbooks that document your routine activities and playbooks that guide your processes for issue resolution. Understand the benefits and risks to make informed decisions to permit changes to enter production.

AWS allows you to view your entire workload (applications, infrastructure, policy, governance, and operations) as code. This means you can apply the same engineering discipline that you use for application code to every element of your stack and share these across teams or organizations to magnify the benefits of development efforts. Use operations as code in the cloud and the ability to safely experiment to develop your workload, your operations procedures, and practice failure. Using CloudFormation allows you to have consistent, templated, sandbox development, test, and production environments with increasing levels of operations control.

The following questions focus on these considerations for operational excellence.

OPS 4: How do you implement observability in your workload?

Implement observability in your workload so that you can understand its state and make data-driven decisions based on business requirements.

OPS 5: How do you reduce defects, ease remediation, and improve flow into production?

Adopt approaches that improve flow of changes into production that achieve refactoring fast feedback on quality, and bug fixing. These accelerate beneficial changes entering productio n, limit issues deployed, and achieve rapid identification and remediation of issues introduced through deployment activities.

OPS 6: How do you mitigate deployment risks?

Adopt approaches that provide fast feedback on quality and achieve rapid recovery from changes that do not have desired outcomes. Using these practices mitigates the impact of issues introduced through the deployment of changes.

OPS 7: How do you know that you are ready to support a workload?

Evaluate the operational readiness of your workload, processes and procedures, and personnel to understand the operational risks related to your workload.

Invest in implementing operations activities as code to maximize the productivity of operations personnel, minimize error rates, and achieve automated responses. Use “pre-mortems” to anticipate failure and create procedures where appropriate. Apply metadata using Resource Tags and AWS Resource Groups following a consistent tagging strategy to achieve identification of your resources. Tag your resources for organization, cost accounting, access controls, and targeting the running of automated operations activities. Adopt deployment practices that take advantage of the elasticity of the cloud to facilitate development activities, and pre-deployment of systems for faster implementations. When you make changes to the checklists you use to evaluate your workloads, plan what you will do with live systems that no longer comply.

Operate

Observability allows you to focus on meaningful data and understand your workload's interactions and output. By concentrating on essential insights and eliminating unnecessary data, you maintain a straightforward approach to understanding workload performance. It's essential not only to collect data but also to interpret it correctly. Define clear baselines, set appropriate alert thresholds, and actively monitor for any deviations. A shift in a key metric, especially when correlated with other data, can pinpoint specific problem areas. With observability, you're better equipped to foresee and address potential challenges, ensuring that your workload operates smoothly and meets business needs.

Successful operation of a workload is measured by the achievement of business and customer outcomes. Define expected outcomes, determine how success will be measured, and identify metrics that will be used in those calculations to determine if your workload and operations are successful. Operational health includes both the health of the workload and the health and success of the operations activities performed in support of the workload (for example, deployment and incident response). Establish metrics baselines for improvement, investigation, and intervention, collect and analyze your metrics, and then validate your understanding of operations success and how it changes over time. Use collected metrics to determine if you are satisfying customer and business needs, and identify areas for improvement.

Efficient and effective management of operational events is required to achieve operational excellence. This applies to both planned and unplanned operational events. Use established runbooks for well-understood events, and use playbooks to aid in investigation and resolution of issues. Prioritize responses to events based on their business and customer impact. Verify that if an alert is raised in response to an event, there is an associated process to run with a specifically identified owner. Define in advance the personnel required to resolve an event and include escalation processes to engage additional personnel, as it becomes necessary, based on urgency and impact. Identify and engage individuals with the authority to make a decision on courses of action where there will be a business impact from an event response not previously addressed.

Communicate the operational status of workloads through dashboards and notifications that are tailored to the target audience (for example, customer, business, developers, operations) so that they may take appropriate action, so that their expectations are managed, and so that they are informed when normal operations resume.

In AWS, you can generate dashboard views of your metrics collected from workloads and natively from AWS. You can leverage CloudWatch or third-party applications to aggregate and present business, workload, and operations level views of operations activities. AWS provides workload insights through logging capabilities including AWS X-Ray, CloudWatch, CloudTrail, and VPC Flow Logs to identify workload issues in support of root cause analysis and remediation.

The following questions focus on these considerations for operational excellence.

OPS 8: How do you utilize workload observability in your organization?

Ensure optimal workload health by leveraging observability. Utilize relevant metrics, logs, and traces to gain a comprehensive view of your workload's performance and address issues efficient ly.

OPS 9: How do you understand the health of your operations?

Define, capture, and analyze operations metrics to gain visibility to operations events so that you can take appropriate action.

OPS 10: How do you manage workload and operations events?

Prepare and validate procedures for responding to events to minimize their disruption to your workload.

All of the metrics you collect should be aligned to a business need and the outcomes they support. Develop scripted responses to well-understood events and automate their performance in response to recognizing the event.

Evolve

Learn, share, and continuously improve to sustain operational excellence. Dedicate work cycles to making nearly continuous incremental improvements. Perform post-incident analysis of all customer impacting events. Identify the contributing factors and preventative action to limit or prevent recurrence. Communicate contributing factors with affected communities as appropriate. Regularly evaluate and prioritize opportunities for improvement (for example, feature requests, issue remediation, and compliance requirements), including both the workload and operations procedures.

Include feedback loops within your procedures to rapidly identify areas for improvement and capture learnings from running operations.

Share lessons learned across teams to share the benefits of those lessons. Analyze trends within lessons learned and perform cross-team retrospective analysis of operations metrics to identify opportunities and methods for improvement. Implement changes intended to bring about improvement and evaluate the results to determine success.

On AWS, you can export your log data to Amazon S3 or send logs directly to Amazon S3 for long-term storage. Using AWS Glue, you can discover and prepare your log data in Amazon S3 for analytics, and store associated metadata in the AWS Glue Data Catalog. Amazon Athena, through its native integration with AWS Glue, can then be used to analyze your log data, querying it using standard SQL. Using a business intelligence tool like Amazon Quick, you can visualize, explore, and analyze your data. Discovering trends and events of interest that may drive improvement.

The following question focuses on these considerations for operational excellence.

OPS 11: How do you evolve operations?

Dedicate time and resources for nearly continuous incremental improvement to evolve the effectiveness and efficiency of your operations.

Successful evolution of operations is founded in: frequent small improvements; providing safe environments and time to experiment, develop, and test improvements; and environments in which learning from failures is encouraged. Operations support for sandbox, development, test, and production environments, with increasing level of operational controls, facilitates development and increases the predictability of successful results from changes deployed into production.

Resources

Refer to the following resources to learn more about our best practices for Operational Excellence.

Documentation

- DevOps and AWS

Whitepaper

- Operational Excellence Pillar

Video

- DevOps at Amazon

Security

The Security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security.

The security pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Security Pillar whitepaper.

Topics

- Design principles

- Definition
- Best practices
- Resources

Design principles

In the cloud, there are a number of principles that can help you strengthen your workload security:

- **Implement a strong identity foundation:** Implement the principle of least privilege and enforce separation of duties with appropriate authorization for each interaction with your AWS resources. Centralize identity management, and aim to eliminate reliance on long-term static credentials.
- **Maintain traceability:** Monitor, alert, and audit actions and changes to your environment in real time. Integrate log and metric collection with systems to automatically investigate and take action.
- **Apply security at all layers:** Apply a defense in depth approach with multiple security controls. Apply to all layers (for example, edge of network, VPC, load balancing, every instance and compute service, operating system, application, and code).
- **Automate security best practices:** Automated software-based security mechanisms improve your ability to securely scale more rapidly and cost-effectively. Create secure architectures, including the implementation of controls that are defined and managed as code in version-controlled templates.
- **Protect data in transit and at rest**: Classify your data into sensitivity levels and use mechanisms, such as encryption, tokenization, and access control where appropriate.
- **Keep people away from data:** Use mechanisms and tools to reduce or eliminate the need for direct access or manual processing of data. This reduces the risk of mishandling or modification and human error when handling sensitive data.
- **Prepare for security events:** Prepare for an incident by having incident management and investigation policy and processes that align to your organizational requirements. Run incident response simulations and use tools with automation to increase your speed for detection, investigation, and recovery.

Definition

There are seven best practice areas for security in the cloud:

- Security foundations
- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

Before you architect any workload, you need to put in place practices that influence security. You will want to control who can do what. In addition, you want to be able to identify security incidents, protect your systems and services, and maintain the confidentiality and integrity of data through data protection. You should have a well-defined and practiced process for responding to security incidents. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

The AWS Shared Responsibility Model helps organizations that adopt the cloud to achieve their security and compliance goals. Because AWS physically secures the infrastructure that supports our cloud services, as an AWS customer you can focus on using services to accomplish your goals. The AWS Cloud also provides greater access to security data and an automated approach to responding to security events.

Best practices

Topics

- Security foundations
- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

Security foundations

The following question focuses on these considerations for security. (For a list of security questions and best practices, see the Appendix.).

SEC 1: How do you securely operate your workload?

To operate your workload securely, you must apply overarching best practices to every area of security. Take requirements and processes that you have defined in operational excellence at an organizational and workload level, and apply them to all areas.

Staying up to date with recommendations from AWS, industry sources, and threat intelligence helps you evolve your threat model and control objectives. Automating security processes, testing, and validation allow you to scale your security operations.

In AWS, segregating different workloads by account, based on their function and compliance or data sensitivity requirements, is a recommended approach.

Identity and access management

Identity and access management are key parts of an information security program, ensuring that only authorized and authenticated users and components are able to access your resources, and only in a manner that you intend. For example, you should define principals (that is, accounts, users, roles, and services that can perform actions in your account), build out policies aligned with these principals, and implement strong credential management. These privilege-management elements form the core of authentication and authorization.

In AWS, privilege management is primarily supported by the AWS Identity and Access Management (IAM) service, which allows you to control user and programmatic access to AWS services and resources. You should apply granular policies, which assign permissions to a user, group, role, or resource. You also have the ability to require strong password practices, such as complexity level, avoiding re-use, and enforcing multi-factor authentication (MFA). You can use federation with your existing directory service. For workloads that require systems to have access to AWS, IAM allows for secure access through roles, instance profiles, identity federation, and temporary credentials.

The following questions focus on these considerations for security.

SEC 2: How do you manage identities for people and machines?

There are two types of identities you need to manage when approaching operating secure AWS workloads. Understanding the type of identity you need to manage and grant access helps you verify the right identities have access to the right resources under the right conditions.

Human Identities: Your administrators, developers, operators, and end users require an identity to access your AWS environments and applications. These are members of your organization, or external users with whom you collaborate, and who interact with your AWS resources via a web browser, client application, or interactive command line tools.

Machine Identities: Your service applications, operational tools, and workloads require an identity to make requests to AWS services, for example, to read data. These identities include machines running in your AWS environment such as Amazon EC2 instances or AWS Lambda functions. You may also manage machine identities for external parties who need access. Additionally, you may also have machines outside of AWS that need access to your AWS environment.

SEC 3: How do you manage permissions for people and machines?

Manage permissions to control access to people and machine identities that require access to AWS and your workload. Permissions control who can access what, and under what conditions.

Credentials must not be shared between any user or system. User access should be granted using a least-privilege approach with best practices including password requirements and MFA enforced. Programmatic access, including API calls to AWS services, should be performed using temporary and limited-privilege credentials, such as those issued by the AWS Security Token Service.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

AWS provides resources that can help you with identity and access management. To help learn best practices, explore our hands-on labs on managing credentials & authentication, controlling human access, and controlling programmatic access.

Detection

You can use detective controls to identify a potential security threat or incident. They are an essential part of governance frameworks and can be used to support a quality process, a legal or compliance obligation, and for threat identification and response efforts. There are different types of detective controls. For example, conducting an inventory of assets and their detailed attributes promotes more effective decision making (and lifecycle controls) to help establish operational baselines. You can also use internal auditing, an examination of controls related to information systems, to verify that practices meet policies and requirements and that you have set the correct automated alerting notifications based on defined conditions. These controls are

important reactive factors that can help your organization identify and understand the scope of anomalous activity.

In AWS, you can implement detective controls by processing logs, events, and monitoring that allows for auditing, automated analysis, and alarming. CloudTrail logs, AWS API calls, and CloudWatch provide monitoring of metrics with alarming, and AWS Config provides configuration history. Amazon GuardDuty is a managed threat detection service that continuously monitors for malicious or unauthorized behavior to help you protect your AWS accounts and workloads. Service-level logs are also available, for example, you can use Amazon Simple Storage Service (Amazon S3) to log access requests.

The following question focuses on these considerations for security.

SEC 4: How do you detect and investigate security events?

Capture and analyze events from logs and metrics to gain visibility. Take action on security events and potential threats to help secure your workload.

Log management is important to a Well-Architected workload for reasons ranging from security or forensics to regulatory or legal requirements. It is critical that you analyze logs and respond to them so that you can identify potential security incidents. AWS provides functionality that makes log management easier to implement by giving you the ability to define a data-retention lifecycle or define where data will be preserved, archived, or eventually deleted. This makes predictable and reliable data handling simpler and more cost effective.

Infrastructure protection

Infrastructure protection encompasses control methodologies, such as defense in depth, necessary to meet best practices and organizational or regulatory obligations. Use of these methodologies is critical for successful, ongoing operations in either the cloud or on-premises.

In AWS, you can implement stateful and stateless packet inspection, either by using AWS-native technologies or by using partner products and services available through the AWS Marketplace. You should use Amazon Virtual Private Cloud (Amazon VPC) to create a private, secured, and scalable environment in which you can define your topology—including gateways, routing tables, and public and private subnets.

The following questions focus on these considerations for security.

SEC 5: How do you protect your network resources?

Any workload that has some form of network connectivity, whether it's the internet or a private network, requires multiple layers of defense to help protect from external and internal network-based threats.

SEC 6: How do you protect your compute resources?

Compute resources in your workload require multiple layers of defense to help protect from external and internal threats. Compute resources include EC2 instances, containers, AWS Lambda functions, database services, IoT devices, and more.

Multiple layers of defense are advisable in any type of environment. In the case of infrastructure protection, many of the concepts and methods are valid across cloud and on-premises models. Enforcing boundary protection, monitoring points of ingress and egress, and comprehensive logging, monitoring, and alerting are all essential to an effective information security plan.

AWS customers are able to tailor, or harden, the configuration of an Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS) container, or AWS Elastic Beanstalk instance, and persist this configuration to an immutable Amazon Machine Image (AMI). Then, whether launched by Auto Scaling or launched manually, all new virtual servers (instances) launched with this AMI receive the hardened configuration.

Data protection

Before architecting any system, foundational practices that influence security should be in place. For example, data classification provides a way to categorize organizational data based on levels of sensitivity, and encryption protects data by way of rendering it unintelligible to unauthorized access. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

In AWS, the following practices facilitate protection of data:

- As an AWS customer you maintain full control over your data.
- AWS makes it easier for you to encrypt your data and manage keys, including regular key rotation, which can be easily automated by AWS or maintained by you.

- Detailed logging that contains important content, such as file access and changes, is available.
- AWS has designed storage systems for exceptional resiliency. For example, Amazon S3 Standard, S3 Standard–IA, S3 One Zone-IA, and Amazon Glacier are all designed to provide 99.999999999% durability of objects over a given year. This durability level corresponds to an average annual expected loss of 0.000000001% of objects.
- Versioning, which can be part of a larger data lifecycle management process, can protect against accidental overwrites, deletes, and similar harm.
- AWS never initiates the movement of data between Regions. Content placed in a Region will remain in that Region unless you explicitly use a feature or leverage a service that provides that functionality.

The following questions focus on these considerations for security.

SEC 7: How do you classify your data?

Classification provides a way to categorize data, based on criticality and sensitivity in order to help you determine appropriate protection and retention controls.

SEC 8: How do you protect your data at rest?

Protect your data at rest by implementing multiple controls, to reduce the risk of unauthorized access or mishandling.

SEC 9: How do you protect your data in transit?

Protect your data in transit by implementing multiple controls to reduce the risk of unauthori zed access or loss.

AWS provides multiple means for encrypting data at rest and in transit. We build features into our services that make it easier to encrypt your data. For example, we have implemented server-side encryption (SSE) for Amazon S3 to make it easier for you to store your data in an encrypted form. You can also arrange for the entire HTTPS encryption and decryption process (generally known as SSL termination) to be handled by Elastic Load Balancing (ELB).

Incident response

Even with extremely mature preventive and detective controls, your organization should still put processes in place to respond to and mitigate the potential impact of security incidents. The architecture of your workload strongly affects the ability of your teams to operate effectively during an incident, to isolate or contain systems, and to restore operations to a known good state. Putting in place the tools and access ahead of a security incident, then routinely practicing incident response through game days, will help you verify that your architecture can accommodate timely investigation and recovery.

In AWS, the following practices facilitate effective incident response:

- Detailed logging is available that contains important content, such as file access and changes.
- Events can be automatically processed and launch tools that automate responses through the use of AWS APIs.
- You can pre-provision tooling and a “clean room” using AWS CloudFormation. This allows you to carry out forensics in a safe, isolated environment.

The following question focuses on these considerations for security.

SEC 10: How do you anticipate, respond to, and recover from incidents?

Preparation is critical to timely and effective investigation, response to, and recovery from security incidents to help minimize disruption to your organization.

Verify that you have a way to quickly grant access for your security team, and automate the isolation of instances as well as the capturing of data and state for forensics.

Application security

Application security (AppSec) describes the overall process of how you design, build, and test the security properties of the workloads you develop. You should have appropriately trained people in your organization, understand the security properties of your build and release infrastructure, and use automation to identify security issues.

Adopting application security testing as a regular part of your software development lifecycle (SDLC) and post release processes help validate that you have a structured mechanism to identify, fix, and prevent application security issues entering your production environment.

Your application development methodology should include security controls as you design, build, deploy, and operate your workloads. While doing so, align the process for continuous defect reduction and minimizing technical debt. For example, using threat modeling in the design phase helps you uncover design flaws early, which makes them easier and less costly to fix as opposed to waiting and mitigating them later.

The cost and complexity to resolve defects is typically lower the earlier you are in the SDLC. The easiest way to resolve issues is to not have them in the first place, which is why starting with a threat model helps you focus on the right outcomes from the design phase. As your AppSec program matures, you can increase the amount of testing that is performed using automation, improve the fidelity of feedback to builders, and reduce the time needed for security reviews. All of these actions improve the quality of the software you build, and increase the speed of delivering features into production.

These implementation guidelines focus on four areas: organization and culture, security *of* the pipeline, security *in* the pipeline, and dependency management. Each area provides a set of principles that you can implement and provides an end-to-end view of how you design, develop, build, deploy, and operate workloads.

In AWS, there are a number of approaches you can use when addressing your application security program. Some of these approaches rely on technology while others focus on the people and organizational aspects of your application security program.

The following question focuses on these considerations for application security.

SEC 11: How do you incorporate and validate the security properties of applications throughout the design, development, and deployment lifecycle?

Training people, testing using automation, understanding dependencies, and validating the security properties of tools and applications help to reduce the likelihood of security issues in production workloads.

Resources

Refer to the following resources to learn more about our best practices for Security.

Documentation

- AWS Cloud Security

- AWS Compliance
- AWS Security Blog
- AWS Security Maturity Model

Whitepaper

- Security Pillar
- AWS Security Overview
- AWS Risk and Compliance

Video

- AWS Security State of the Union
- Shared Responsibility Overview

Reliability

The Reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle. This paper provides in-depth, best practice guidance for implementing reliable workloads on AWS.

The reliability pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Reliability Pillar whitepaper.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

There are five design principles for reliability in the cloud:

- **Automatically recover from failure**: By monitoring a workload for key performance indicators (KPIs), you can start automation when a threshold is breached. These KPIs should be a measure of business value, not of the technical aspects of the operation of the service. This provides for automatic notification and tracking of failures, and for automated recovery processes that work around or repair the failure. With more sophisticated automation, it's possible to anticipate and remediate failures before they occur.
- **Test recovery procedures**: In an on-premises environment, testing is often conducted to prove that the workload works in a particular scenario. Testing is not typically used to validate recovery strategies. In the cloud, you can test how your workload fails, and you can validate your recovery procedures. You can use automation to simulate different failures or to recreate scenarios that led to failures before. This approach exposes failure pathways that you can test and fix before a real failure scenario occurs, thus reducing risk.
- **Scale horizontally to increase aggregate workload availability**: Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall workload. Distribute requests across multiple, smaller resources to verify that they don't share a common point of failure.
- **Stop guessing capacity**: A common cause of failure in on-premises workloads is resource saturation, when the demands placed on a workload exceed the capacity of that workload (this is often the objective of denial of service attacks). In the cloud, you can monitor demand and workload utilization, and automate the addition or removal of resources to maintain the more efficient level to satisfy demand without over- or under-provisioning. There are still limits, but some quotas can be controlled and others can be managed (see Manage Service Quotas and Constraints).
- **Manage change through automation**: Changes to your infrastructure should be made using automation. The changes that must be managed include changes to the automation, which then can be tracked and reviewed.

Definition

There are four best practice areas for reliability in the cloud:

- Foundations
- Workload architecture
- Change management
- Failure management

To achieve reliability, you must start with the foundations — an environment where Service Quotas and network topology accommodate the workload. The workload architecture of the distributed system must be designed to prevent and mitigate failures. The workload must handle changes in demand or requirements, and it must be designed to detect failure and automatically heal itself.

Best practices

Topics

- Foundations
- Workload architecture
- Change management
- Failure management

Foundations

Foundational requirements are those whose scope extends beyond a single workload or project. Before architecting any system, foundational requirements that influence reliability should be in place. For example, you must have sufficient network bandwidth to your data center.

With AWS, most of these foundational requirements are already incorporated or can be addressed as needed. The cloud is designed to be nearly limitless, so it's the responsibility of AWS to satisfy the requirement for sufficient networking and compute capacity, permitting you to change resource size and allocations on demand.

The following questions focus on these considerations for reliability. (For a list of reliability questions and best practices, see the Appendix.).

REL 1: How do you manage Service Quotas and constraints?

For cloud-based workload architectures, there are Service Quotas (which are also referred to as service limits). These quotas exist to prevent accidentally provisioning more resources than you need and to limit request rates on API operations so as to protect services from abuse. There are also resource constraints, for example, the rate that you can push bits down a fiber-optic cable, or the amount of storage on a physical disk.

REL 2: How do you plan your network topology?

Workloads often exist in multiple environments. These include multiple cloud environments (both publicly accessible and private) and possibly your existing data center infrastructure. Plans must include network considerations such as intra- and inter-system connectivity, public IP address management, private IP address management, and domain name resolution.

Workload architecture

A reliable workload starts with upfront design decisions for both software and infrastructure. Your architecture choices will impact your workload behavior across all of the Well-Architected pillars. For reliability, there are specific patterns you must follow.

With AWS, workload developers have their choice of languages and technologies to use. AWS SDKs take the complexity out of coding by providing language-specific APIs for AWS services. These SDKs, plus the choice of languages, permits developers to implement the reliability best practices listed here. Developers can also read about and learn from how Amazon builds and operates software in The Amazon Builders' Library.

The following questions focus on these considerations for reliability.

REL 3: How do you design your workload service architecture?

Build highly scalable and reliable workloads using a service-oriented architecture (SOA) or a microservices architecture. Service-oriented architecture (SOA) is the practice of making software components reusable via service interfaces. Microservices architecture goes further to make components smaller and simpler.

REL 4: How do you design interactions in a distributed system to prevent failures?

Distributed systems rely on communications networks to interconnect components, such as servers or services. Your workload must operate reliably despite data loss or latency in these networks. Components of the distributed system must operate in a way that does not negativel y impact other components or the workload. These best practices prevent failures and improve mean time between failures (MTBF).

REL 5: How do you design interactions in a distributed system to mitigate or withstand failures?

Distributed systems rely on communications networks to interconnect components (such as servers or services). Your workload must operate reliably despite data loss or latency over these networks. Components of the distributed system must operate in a way that does not negatively impact other components or the workload. These best practices permit workloads to withstand stresses or failures, more quickly recover from them, and mitigate the impact of such impairmen ts. The result is improved mean time to recovery (MTTR).

Change management

Changes to your workload or its environment must be anticipated and accommodated to achieve reliable operation of the workload. Changes include those imposed on your workload, such as spikes in demand, and also those from within, such as feature deployments and security patches.

Using AWS, you can monitor the behavior of a workload and automate the response to KPIs. For example, your workload can add additional servers as a workload gains more users. You can control who has permission to make workload changes and audit the history of these changes.

The following questions focus on these considerations for reliability.

REL 6: How do you monitor workload resources?

Logs and metrics are powerful tools to gain insight into the health of your workload. You can configure your workload to monitor logs and metrics and send notifications when thresholds are crossed or significant events occur. Monitoring allows your workload to recognize when low-performance thresholds are crossed or failures occur, so it can recover automatically in response.

REL 7: How do you design your workload to adapt to changes in demand?

A scalable workload provides elasticity to add or remove resources automatically so that they closely match the current demand at any given point in time.

REL 8: How do you implement change?

Controlled changes are necessary to deploy new functionality, and to verify that the workloads and the operating environment are running known software and can be patched or replaced in a predictable manner. If these changes are uncontrolled, then it makes it difficult to predict the effect of these changes, or to address issues that arise because of them.

When you architect a workload to automatically add and remove resources in response to changes in demand, this not only increases reliability but also validates that business success doesn't become a burden. With monitoring in place, your team will be automatically alerted when KPIs deviate from expected norms. Automatic logging of changes to your environment permits you to audit and quickly identify actions that might have impacted reliability. Controls on change management certify that you can enforce the rules that deliver the reliability you need.

Failure management

In any system of reasonable complexity, it is expected that failures will occur. Reliability requires that your workload be aware of failures as they occur and take action to avoid impact on availability. Workloads must be able to both withstand failures and automatically repair issues.

With AWS, you can take advantage of automation to react to monitoring data. For example, when a particular metric crosses a threshold, you can initiate an automated action to remedy the problem. Also, rather than trying to diagnose and fix a failed resource that is part of your production environment, you can replace it with a new one and carry out the analysis on the failed resource out of band. Since the cloud allows you to stand up temporary versions of a whole system at low cost, you can use automated testing to verify full recovery processes.

The following questions focus on these considerations for reliability.

REL 9: How do you back up data?

Back up data, applications, and configuration to meet your requirements for recovery time objectives (RTO) and recovery point objectives (RPO).

REL 10: How do you use fault isolation to protect your workload?

Fault isolation limits the impact of a component or system failure to a defined boundary. With proper isolation, components outside of the boundary are unaffected by the failure. Running your workload across multiple fault isolation boundaries can make it more resilient to failure.

REL 11: How do you design your workload to withstand component failures?

Workloads with a requirement for high availability and low mean time to recovery (MTTR) must be architected for resiliency.

REL 12: How do you test reliability?

After you have designed your workload to be resilient to the stresses of production, testing is the only way to verify that it will operate as designed, and deliver the resiliency you expect.

REL 13: How do you plan for disaster recovery (DR)?

Having backups and redundant workload components in place is the start of your DR strategy. RTO and RPO are your objectives for restoration of your workload. Set these based on business needs. Implement a strategy to meet these objectives, considering locations and function of workload resources and data. The probability of disruption and cost of recovery are also key factors that help to inform the business value of providing disaster recovery for a workload.

Regularly back up your data and test your backup files to verify that you can recover from both logical and physical errors. A key to managing failure is the frequent and automated testing of workloads to cause failure, and then observe how they recover. Do this on a regular schedule and verify that such testing is also initiated after significant workload changes. Actively track KPIs, and also the recovery time objective (RTO) and recovery point objective (RPO), to assess a workload's resiliency (especially under failure-testing scenarios). Tracking KPIs will help you identify and mitigate single points of failure. The objective is to thoroughly test your workload-recovery processes so that you are confident that you can recover all your data and continue to serve your

customers, even in the face of sustained problems. Your recovery processes should be as well exercised as your normal production processes.

Resources

Refer to the following resources to learn more about our best practices for Reliability.

Documentation

- AWS Documentation
- AWS Global Infrastructure
- AWS Auto Scaling: How Scaling Plans Work
- What Is AWS Backup?

Whitepaper

- Reliability Pillar: AWS Well-Architected
- Implementing Microservices on AWS

Performance efficiency

The performance efficiency pillar includes the ability to use cloud resources efficiently to meet performance requirements, and to maintain that efficiency as demand changes and technologies evolve.

The performance efficiency pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Performance Efficiency Pillar whitepaper.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

There are five design principles for performance efficiency in the cloud:

- **Democratize advanced technologies**: Make advanced technology implementation smoother for your team by delegating complex tasks to your cloud vendor. Rather than asking your IT team to learn about hosting and running a new technology, consider consuming the technology as a service. For example, NoSQL databases, media transcoding, and machine learning are all technologies that require specialized expertise. In the cloud, these technologies become services that your team can consume, permitting your team to focus on product development rather than resource provisioning and management.
- **Go global in minutes**: Deploying your workload in multiple AWS Regions around the world permits you to provide lower latency and a better experience for your customers at minimal cost.
- **Use serverless architectures**: Serverless architectures remove the need for you to run and maintain physical servers for traditional compute activities. For example, serverless storage services can act as static websites (removing the need for web servers) and event services can host code. This removes the operational burden of managing physical servers, and can lower transactional costs because managed services operate at cloud scale.
- **Experiment more often**: With virtual and automatable resources, you can quickly carry out comparative testing using different types of instances, storage, or configurations.
- **Consider mechanical sympathy**: Understand how cloud services are consumed and always use the technology approach that aligns with your workload goals. For example, consider data access patterns when you select database or storage approaches.

Definition

There are five best practice areas for performance efficiency in the cloud:

- **Architecture selection**
- **Compute and hardware**
- **Data management**
- **Networking and content delivery**
- **Process and culture**

Take a data-driven approach to building a high-performance architecture. Gather data on all aspects of the architecture, from the high-level design to the selection and configuration of resource types.

Reviewing your choices on a regular basis validates that you are taking advantage of the continually evolving AWS Cloud. Monitoring verifies that you are aware of any deviance from expected performance. Make trade-offs in your architecture to improve performance, such as using compression or caching, or relaxing consistency requirements.

Best practices

Topics

- Architecture selection
- Compute and hardware
- Data management
- Networking and content delivery
- Process and culture

Architecture selection

The optimal solution for a particular workload varies, and solutions often combine multiple approaches. Well-Architected workloads use multiple solutions and allow different features to improve performance.

AWS resources are available in many types and configurations, which makes it easier to find an approach that closely matches your needs. You can also find options that are not easily achievable with on-premises infrastructure. For example, a managed service such as Amazon DynamoDB provides a fully managed NoSQL database with single-digit millisecond latency at any scale.

The following question focuses on these considerations for performance efficiency. (For a list of performance efficiency questions and best practices, see the Appendix.).

PERF 1: How do you select appropriate cloud resources and architecture patterns for your workload?

Often, multiple approaches are required for more effective performance across a workload. Well-Architected systems use multiple solutions and features to improve performance.

Compute and hardware

The optimal compute choice for a particular workload can vary based on application design, usage patterns, and configuration settings. Architectures may use different compute choices for various components and allow different features to improve performance. Selecting the wrong compute choice for an architecture can lead to lower performance efficiency.

In AWS, compute is available in three forms: instances, containers, and functions:

- **Instances** are virtualized servers, permitting you to change their capabilities with a button or an API call. Because resource decisions in the cloud aren't fixed, you can experiment with different server types. At AWS, these virtual server instances come in different families and sizes, and they offer a wide variety of capabilities, including solid-state drives (SSDs) and graphics processing units (GPUs).
- **Containers** are a method of operating system virtualization that permit you to run an application and its dependencies in resource-isolated processes. AWS Fargate is serverless compute for containers or Amazon EC2 can be used if you need control over the installation, configuration, and management of your compute environment. You can also choose from multiple container orchestration platforms: Amazon Elastic Container Service (ECS) or Amazon Elastic Kubernetes Service (EKS).
- **Functions** abstract the run environment from the code you want to apply. For example, AWS Lambda permits you to run code without running an instance.

The following question focuses on these considerations for performance efficiency.

PERF 2: How do you select and use compute resources in your workload?

The more efficient compute solution for a workload varies based on application design, usage patterns, and configuration settings. Architectures can use different compute solutions for various components and turn on different features to improve performance. Selecting the wrong compute solution for an architecture can lead to lower performance efficiency.

Data management

The optimal data management solution for a particular system varies based on the kind of data type (block, file, or object), access patterns (random or sequential), required throughput, frequency

of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-Architected workloads use purpose-built data stores which allow different features to improve performance.

In AWS, storage is available in three forms: object, block, and file:

- **Object storage** provides a scalable, durable platform to make data accessible from any internet location for user-generated content, active archive, serverless computing, Big Data storage or backup and recovery. Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Amazon S3 is designed for 99.999999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world.
- **Block storage** provides highly available, consistent, low-latency block storage for each virtual host and is analogous to direct-attached storage (DAS) or a Storage Area Network (SAN). Amazon Elastic Block Store (Amazon EBS) is designed for workloads that require persistent storage accessible by EC2 instances that helps you tune applications with the right storage capacity, performance and cost.
- **File storage** provides access to a shared file system across multiple systems. File storage solutions like Amazon Elastic File System (Amazon EFS) are ideal for use cases such as large content repositories, development environments, media stores, or user home directories. Amazon FSx makes it efficient and cost effective to launch and run popular file systems so you can leverage the rich feature sets and fast performance of widely used open source and commercially-licensed file systems.

The following question focuses on these considerations for performance efficiency.

PERF 3: How do you store, manage, and access data in your workload?

The more efficient storage solution for a system varies based on the kind of access operation (block, file, or object), patterns of access (random or sequential), required throughput, frequency of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-architected systems use multiple storage solutions and turn on different features to improve performance and use resources efficiently.

Networking and content delivery

The optimal networking solution for a workload varies based on latency, throughput requirements, jitter, and bandwidth. Physical constraints, such as user or on-premises resources, determine location options. These constraints can be offset with edge locations or resource placement.

On AWS, networking is virtualized and is available in a number of different types and configurations. This makes it easier to match your networking needs. AWS offers product features (for example, Enhanced Networking, Amazon EC2 networking optimized instances, Amazon S3 transfer acceleration, and dynamic Amazon CloudFront) to optimize network traffic. AWS also offers networking features (for example, Amazon Route 53 latency routing, Amazon VPC endpoints, AWS Direct Connect, and AWS Global Accelerator) to reduce network distance or jitter.

The following question focuses on these considerations for performance efficiency.

PERF 4: How do you select and configure networking resources in your workload?

This question includes guidance and best practices to design, configure, and operate efficient networking and content delivery solutions in the cloud.

Process and culture

When architecting workloads, there are principles and practices that you can adopt to help you better run efficient high-performing cloud workloads. To adopt a culture that fosters performance efficiency of cloud workloads, consider these key principles and practices.

Consider these key principles to build this culture:

- **Infrastructure as code:** Define your infrastructure as code using approaches such as AWS CloudFormation templates. The use of templates allows you to place your infrastructure into source control alongside your application code and configurations. This allows you to apply the same practices you use to develop software in your infrastructure so you can iterate rapidly.
- **Deployment pipeline:** Use a continuous integration/continuous deployment (CI/CD) pipeline (for example, source code repository, build systems, deployment, and testing automation) to deploy your infrastructure. This allows you to deploy in a repeatable, consistent, and low-cost fashion as you iterate.
- **Well-defined metrics:** Set up and monitor metrics to capture key performance indicators (KPIs). We recommend that you use both technical and business metrics. For websites or mobile apps,

key metrics are capturing time-to-first-byte or rendering. Other generally applicable metrics include thread count, garbage collection rate, and wait states. Business metrics, such as the aggregate cumulative cost per request, can alert you to ways to drive down costs. Carefully consider how you plan to interpret metrics. For example, you could choose the maximum or 99th percentile instead of the average.

- **Performance test automatically:** As part of your deployment process, automatically start performance tests after the quicker running tests have passed successfully. The automation should create a new environment, set up initial conditions such as test data, and then run a series of benchmarks and load tests. Results from these tests should be tied back to the build so you can track performance changes over time. For long-running tests, you can make this part of the pipeline asynchronous from the rest of the build. Alternatively, you could run performance tests overnight using Amazon EC2 Spot Instances.
- **Load generation:** You should create a series of test scripts that replicate synthetic or prerecorded user journeys. These scripts should be idempotent and not coupled, and you might need to include *pre-warming* scripts to yield valid results. As much as possible, your test scripts should replicate the behavior of usage in production. You can use software or software-as-a-service (SaaS) solutions to generate the load. Consider using AWS Marketplace solutions and Spot Instances — they can be cost-effective ways to generate the load.
- **Performance visibility:** Key metrics should be visible to your team, especially metrics against each build version. This allows you to see any significant positive or negative trend over time. You should also display metrics on the number of errors or exceptions to make sure you are testing a working system.
- **Visualization:** Use visualization techniques that make it clear where performance issues, hot spots, wait states, or low utilization is occurring. Overlay performance metrics over architecture diagrams — call graphs or code can help identify issues quickly.
- **Regular review process:** Architectures performing poorly is usually the result of a non-existent or broken performance review process. If your architecture is performing poorly, implementing a performance review process allows you to drive iterative improvement.
- **Continual optimization:** Adopt a culture to continually optimize the performance efficiency of your cloud workload.

The following question focuses on these considerations for performance efficiency.

PERF 5: What process do you use to support more performance efficiency for your workload?

When architecting workloads, there are principles and practices that you can adopt to help you better run efficient high-performing cloud workloads. To adopt a culture that fosters performan ce efficiency of cloud workloads, consider these key principles and practices.

Resources

Refer to the following resources to learn more about our best practices for Performance Efficiency.

Documentation

- Amazon S3 Performance Optimization
- Amazon EBS Volume Performance

Whitepaper

- Performance Efficiency Pillar

Video

- AWS re:Invent 2019: Amazon EC2 foundations (CMP211-R2)
- AWS re:Invent 2019: Leadership session: Storage state of the union (STG201-L)
- AWS re:Invent 2019: Leadership session: AWS purpose-built databases (DAT209-L)
- AWS re:Invent 2019: Connectivity to AWS and hybrid AWS network architectures (NET317-R1)
- AWS re:Invent 2019: Powering next-gen Amazon EC2: Deep dive into the Nitro system (CMP303-R2)
- AWS re:Invent 2019: Scaling up to your first 10 million users (ARC211-R)

Cost optimization

The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point.

The cost optimization pillar provides an overview of design principles, best practices, and questions. You can find prescriptive guidance on implementation in the Cost Optimization Pillar whitepaper.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

There are five design principles for cost optimization in the cloud:

- **Implement Cloud Financial Management**: To achieve financial success and accelerate business value realization in the cloud, invest in Cloud Financial Management and Cost Optimization. Your organization should dedicate time and resources to build capability in this new domain of technology and usage management. Similar to your Security or Operational Excellence capability, you need to build capability through knowledge building, programs, resources, and processes to become a cost-efficient organization.
- **Adopt a consumption model**: Pay only for the computing resources that you require and increase or decrease usage depending on business requirements, not by using elaborate forecasting. For example, development and test environments are typically only used for eight hours a day during the work week. You can stop these resources when they are not in use for a potential cost savings of 75% (40 hours versus 168 hours).
- **Measure overall efficiency**: Measure the business output of the workload and the costs associated with delivering it. Use this measure to know the gains you make from increasing output and reducing costs.
- **Stop spending money on undifferentiated heavy lifting**: AWS does the heavy lifting of data center operations like racking, stacking, and powering servers. It also removes the operational burden of managing operating systems and applications with managed services. This permits you to focus on your customers and business projects rather than on IT infrastructure.
- **Analyze and attribute expenditure**: The cloud makes it simple to accurately identify the usage and cost of systems, which then permits transparent attribution of IT costs to individual

workload owners. This helps measure return on investment (ROI) and gives workload owners an opportunity to optimize their resources and reduce costs.

Definition

There are five best practice areas for cost optimization in the cloud:

- **Practice Cloud Financial Management**
- **Expenditure and usage awareness**
- **Cost-effective resources**
- **Manage demand and supply resources**
- **Optimize over time**

As with the other pillars within the Well-Architected Framework, there are tradeoffs to consider, for example, whether to optimize for speed-to-market or for cost. In some cases, it's more efficient to optimize for speed, going to market quickly, shipping new features, or meeting a deadline, rather than investing in upfront cost optimization. Design decisions are sometimes directed by haste rather than data, and the temptation always exists to overcompensate "just in case" rather than spend time benchmarking for the most cost-optimal deployment. This might lead to over-provisioned and under-optimized deployments. However, this is a reasonable choice when you must "lift and shift" resources from your on-premises environment to the cloud and then optimize afterwards. Investing the right amount of effort in a cost optimization strategy up front permits you to realize the economic benefits of the cloud more readily by achieving a consistent adherence to best practices and avoiding unnecessary over provisioning. The following sections provide techniques and best practices for both the initial and ongoing implementation of Cloud Financial Management and cost optimization of your workloads.

Best practices

Topics

- Practice Cloud Financial Management
- Expenditure and usage awareness
- Cost-effective resources
- Manage demand and supply resources
- Optimize over time

Practice Cloud Financial Management

With the adoption of cloud, technology teams innovate faster due to shortened approval, procurement, and infrastructure deployment cycles. A new approach to financial management in the cloud is required to realize business value and financial success. This approach is Cloud Financial Management, and builds capability across your organization by implementing organizational wide knowledge building, programs, resources, and processes.

Many organizations are composed of many different units with different priorities. The ability to align your organization to an agreed set of financial objectives, and provide your organization the mechanisms to meet them, will create a more efficient organization. A capable organization will innovate and build faster, be more agile and adjust to any internal or external factors.

In AWS you can use Cost Explorer, and optionally Amazon Athena and Amazon QuickSight with the Cost and Usage Report (CUR), to provide cost and usage awareness throughout your organization. AWS Budgets provides proactive notifications for cost and usage. The AWS blogs provide information on new services and features to verify you keep up to date with new service releases.

The following question focuses on these considerations for cost optimization. (For a list of cost optimization questions and best practices, see the Appendix.).

COST 1: How do you implement cloud financial management?

Implementing Cloud Financial Management helps organizations realize business value and financial success as they optimize their cost and usage and scale on AWS.

When building a cost optimization function, use members and supplement the team with experts in CFM and cost optimization. Existing team members will understand how the organization currently functions and how to rapidly implement improvements. Also consider including people with supplementary or specialist skill sets, such as analytics and project management.

When implementing cost awareness in your organization, improve or build on existing programs and processes. It is much faster to add to what exists than to build new processes and programs. This will result in achieving outcomes much faster.

Expenditure and usage awareness

The increased flexibility and agility that the cloud provides encourages innovation and fast-paced development and deployment. It decreases the manual processes and time associated with provisioning on-premises infrastructure, including identifying hardware specifications, negotiating price quotations, managing purchase orders, scheduling shipments, and then deploying the resources. However, the ease of use and virtually unlimited on-demand capacity requires a new way of thinking about expenditures.

Many businesses are composed of multiple systems run by various teams. The capability to attribute resource costs to the individual organization or product owners drives efficient usage behavior and helps reduce waste. Accurate cost attribution permits you to know which products are truly profitable, and permits you to make more informed decisions about where to allocate budget.

In AWS, you create an account structure with AWS Organizations or AWS Control Tower, which provides separation and assists in allocation of your costs and usage. You can also use resource tagging to apply business and organization information to your usage and cost. Use AWS Cost Explorer for visibility into your cost and usage, or create customized dashboards and analytics with Amazon Athena and Amazon QuickSight. Controlling your cost and usage is done by notifications through AWS Budgets, and controls using AWS Identity and Access Management (IAM), and Service Quotas.

The following questions focus on these considerations for cost optimization.

COST 2: How do you govern usage?

Establish policies and mechanisms to validate that appropriate costs are incurred while objective s are achieved. By employing a checks-and-balances approach, you can innovate without overspending.

COST 3: How do you monitor usage and cost?

Establish policies and procedures to monitor and appropriately allocate your costs. This permits you to measure and improve the cost efficiency of this workload.

COST 4: How do you decommission resources?

Implement change control and resource management from project inception to end-of-life. This facilitates shutting down unused resources to reduce waste.

You can use cost allocation tags to categorize and track your AWS usage and costs. When you apply tags to your AWS resources (such as EC2 instances or S3 buckets), AWS generates a cost and usage report with your usage and your tags. You can apply tags that represent organization categories (such as cost centers, workload names, or owners) to organize your costs across multiple services.

Verify that you use the right level of detail and granularity in cost and usage reporting and monitoring. For high level insights and trends, use daily granularity with AWS Cost Explorer. For deeper analysis and inspection use hourly granularity in AWS Cost Explorer, or Amazon Athena and Amazon Quick with the Cost and Usage Report (CUR) at an hourly granularity.

Combining tagged resources with entity lifecycle tracking (employees, projects) makes it possible to identify orphaned resources or projects that are no longer generating value to the organization and should be decommissioned. You can set up billing alerts to notify you of predicted overspending.

Cost-effective resources

Using the appropriate instances and resources for your workload is key to cost savings. For example, a reporting process might take five hours to run on a smaller server but one hour to run on a larger server that is twice as expensive. Both servers give you the same outcome, but the smaller server incurs more cost over time.

A well-architected workload uses the most cost-effective resources, which can have a significant and positive economic impact. You also have the opportunity to use managed services to reduce costs. For example, rather than maintaining servers to deliver email, you can use a service that charges on a per-message basis.

AWS offers a variety of flexible and cost-effective pricing options to acquire instances from Amazon EC2 and other services in a way that more effectively fits your needs. *On-Demand Instances* permit you to pay for compute capacity by the hour, with no minimum commitments required. *Savings Plans and Reserved Instances* offer savings of up to 75% off On-Demand pricing. With Spot Instances, you can leverage unused Amazon EC2 capacity and offer savings of up to 90% off On-

Demand pricing. *Spot Instances* are appropriate where the system can tolerate using a fleet of servers where individual servers can come and go dynamically, such as stateless web servers, batch processing, or when using HPC and big data.

Appropriate service selection can also reduce usage and costs; such as CloudFront to minimize data transfer, or decrease costs, such as utilizing Amazon Aurora on Amazon RDS to remove expensive database licensing costs.

The following questions focus on these considerations for cost optimization.

COST 5: How do you evaluate cost when you select services?

Amazon EC2, Amazon EBS, and Amazon S3 are building-block AWS services. Managed services, such as Amazon RDS and Amazon DynamoDB, are higher level, or application level, AWS services. By selecting the appropriate building blocks and managed services, you can optimize this workload for cost. For example, using managed services, you can reduce or remove much of your administrative and operational overhead, freeing you to work on applications and business-related activities.

COST 6: How do you meet cost targets when you select resource type, size and number?

Verify that you choose the appropriate resource size and number of resources for the task at hand. You minimize waste by selecting the most cost effective type, size, and number.

COST 7: How do you use pricing models to reduce cost?

Use the pricing model that is most appropriate for your resources to minimize expense.

COST 8: How do you plan for data transfer charges?

Verify that you plan and monitor data transfer charges so that you can make architectural decisions to minimize costs. A small yet effective architectural change can drastically reduce your operational costs over time.

By factoring in cost during service selection, and using tools such as Cost Explorer and AWS Trusted Advisor to regularly review your AWS usage, you can actively monitor your utilization and adjust your deployments accordingly.

Manage demand and supply resources

When you move to the cloud, you pay only for what you need. You can supply resources to match the workload demand at the time they're needed, this decreases the need for costly and wasteful over provisioning. You can also modify the demand, using a throttle, buffer, or queue to smooth the demand and serve it with less resources resulting in a lower cost, or process it at a later time with a batch service.

In AWS, you can automatically provision resources to match the workload demand. Auto Scaling using demand or time-based approaches permit you to add and remove resources as needed. If you can anticipate changes in demand, you can save more money and validate that your resources match your workload needs. You can use Amazon API Gateway to implement throttling, or Amazon SQS to implementing a queue in your workload. These will both permit you to modify the demand on your workload components.

The following question focuses on these considerations for cost optimization.

COST 9: How do you manage demand, and supply resources?

For a workload that has balanced spend and performance, verify that everything you pay for is used and avoid significantly underutilizing instances. A skewed utilization metric in either direction has an adverse impact on your organization, in either operational costs (degraded performance due to over-utilization), or wasted AWS expenditures (due to over-provisioning).

When designing to modify demand and supply resources, actively think about the patterns of usage, the time it takes to provision new resources, and the predictability of the demand pattern. When managing demand, verify you have a correctly sized queue or buffer, and that you are responding to workload demand in the required amount of time.

Optimize over time

As AWS releases new services and features, it's a best practice to review your existing architectural decisions to verify they continue to be the most cost effective. As your requirements change, be aggressive in decommissioning resources, entire services, and systems that you no longer require.

Implementing new features or resource types can optimize your workload incrementally, while minimizing the effort required to implement the change. This provides continual improvements in efficiency over time and provides you remain on the most updated technology to reduce operating costs. You can also replace or add new components to the workload with new services. This can provide significant increases in efficiency, so it's essential to regularly review your workload, and implement new services and features.

The following questions focus on these considerations for cost optimization.

COST 10: How do you evaluate new services?

As AWS releases new services and features, it's a best practice to review your existing architect ural decisions to verify they continue to be the most cost effective.

When regularly reviewing your deployments, assess how newer services can help save you money. For example, Amazon Aurora on Amazon RDS can reduce costs for relational databases. Using serverless such as Lambda can remove the need to operate and manage instances to run code.

COST 11: How do you evaluate the cost of effort?

Evaluate the cost of effort for operations in the cloud, review your time-consuming cloud operations, and automate them to reduce human efforts and cost by adopting related AWS services, third-party products, or custom tools.

Resources

Refer to the following resources to learn more about our best practices for Cost Optimization.

Documentation

- AWS Documentation

Whitepaper

- Cost Optimization Pillar

Sustainability

The Sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage. You can find prescriptive guidance on implementation in the Sustainability Pillar whitepaper.

Topics

- Design principles
- Definition
- Best practices
- Resources

Design principles

There are six design principles for sustainability in the cloud:

- **Understand your impact:** Measure the impact of your cloud workload and model the future impact of your workload. Include all sources of impact, including impacts resulting from customer use of your products, and impacts resulting from their eventual decommissioning and retirement. Compare the productive output with the total impact of your cloud workloads by reviewing the resources and emissions required per unit of work. Use this data to establish key performance indicators (KPIs), evaluate ways to improve productivity while reducing impact, and estimate the impact of proposed changes over time.
- **Establish sustainability goals:** For each cloud workload, establish long-term sustainability goals such as reducing the compute and storage resources required per transaction. Model the return on investment of sustainability improvements for existing workloads, and give owners the resources they must invest in sustainability goals. Plan for growth, and architect your workloads so that growth results in reduced impact intensity measured against an appropriate unit, such as per user or per transaction. Goals help you support the wider sustainability goals of your business or organization, identify regressions, and prioritize areas of potential improvement.
- **Maximize utilization:** Right-size workloads and implement efficient design to verify high utilization and maximize the energy efficiency of the underlying hardware. Two hosts running at 30% utilization are less efficient than one host running at 60% due to baseline power consumption per host. At the same time, reduce or minimize idle resources, processing, and storage to reduce the total energy required to power your workload.

- **Anticipate and adopt new, more efficient hardware and software offerings:** Support the upstream improvements your partners and suppliers make to help you reduce the impact of your cloud workloads. Continually monitor and evaluate new, more efficient hardware and software offerings. Design for flexibility to permit the rapid adoption of new efficient technologies.
- **Use managed services:** Sharing services across a broad customer base helps maximize resource utilization, which reduces the amount of infrastructure needed to support cloud workloads. For example, customers can share the impact of common data center components like power and networking by migrating workloads to the AWS Cloud and adopting managed services, such as AWS Fargate for serverless containers, where AWS operates at scale and is responsible for their efficient operation. Use managed services that can help minimize your impact, such as automatically moving infrequently accessed data to cold storage with Amazon S3 Lifecycle configurations or Amazon EC2 Auto Scaling to adjust capacity to meet demand.
- **Reduce the downstream impact of your cloud workloads:** Reduce the amount of energy or resources required to use your services. Reduce the need for customers to upgrade their devices to use your services. Test using device farms to understand expected impact and test with customers to understand the actual impact from using your services.

Definition

There are six best practice areas for sustainability in the cloud:

- Region selection
- Alignment to demand
- Software and architecture
- Data
- Hardware and services
- Process and culture

Sustainability in the cloud is a nearly continuous effort focused primarily on energy reduction and efficiency across all components of a workload by achieving the maximum benefit from the resources provisioned and minimizing the total resources required. This effort can range from the initial selection of an efficient programming language, adoption of modern algorithms, use of efficient data storage techniques, deploying to correctly sized and efficient compute infrastructure, and minimizing requirements for high-powered end user hardware.

Best practices

Topics

- Region selection
- Alignment to demand
- Software and architecture
- Data management
- Hardware and services
- Process and culture

Region selection

The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To improve these KPIs, you should choose Regions for your workloads based on both business requirements and sustainability goals.

The following question focuses on these considerations for sustainability. (For a list of sustainability questions and best practices, see the Appendix.)

SUS 1: How do you select Regions for your workload?

The choice of Region for your workload significantly affects its KPIs, including performan ce, cost, and carbon footprint. To improve these KPIs, you should choose Regions for your workloads based on both business requirements and sustainability goals.

Alignment to demand

The way users and applications consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match demand and verify that you use only the minimum resources required to support your users. Align service levels to customer needs. Position resources to limit the network required for users and applications to consume them. Remove unused assets. Provide your team members with devices that support their needs and minimize their sustainability impact.

The following question focuses on this consideration for sustainability:

SUS 2: How do you align cloud resources to your demand?

The way users and applications consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match demand and verify that you use only the minimum resources required to support your users. Align service levels to customer needs. Position resources to limit the network required for users and applications to consume them. Remove unused assets. Provide your team members with devices that support their needs and minimize their sustainability impact.

Scale infrastructure with user load: Identify periods of low or no utilization and scale resources to reduce excess capacity and improve efficiency.

Align SLAs with sustainability goals: Define and update service level agreements (SLAs) such as availability or data retention periods to minimize the number of resources required to support your workload while continuing to meet business requirements.

Decrease creation and maintenance of unused assets: Analyze application assets (such as pre-compiled reports, datasets, and static images) and asset access patterns to identify redundancy, underutilization, and potential decommission targets. Consolidate generated assets with redundant content (for example, monthly reports with overlapping or common datasets and outputs) to reduce the resources consumed when duplicating outputs. Decommission unused assets (for example, images of products that are no longer sold) to release consumed resources and reduce the number of resources used to support the workload.

Optimize geographic placement of workloads for user locations: Analyze network access patterns to identify where your customers are connecting from geographically. Select Regions and services that reduce the distance that network traffic must travel to decrease the total network resources required to support your workload.

Optimize team member resources for activities performed: Optimize resources provided to team members to minimize the sustainability impact while supporting their needs. For example, perform complex operations, such as rendering and compilation, on highly used shared cloud desktops instead of on under-utilized high-powered single user systems.

Software and architecture

Implement patterns for performing load smoothing and maintaining consistent high utilization of deployed resources to minimize the resources consumed. Components might become idle from

lack of use because of changes in user behavior over time. Revise patterns and architecture to consolidate under-utilized components to increase overall utilization. Retire components that are no longer required. Understand the performance of your workload components, and optimize the components that consume the most resources. Be aware of the devices that your customers use to access your services, and implement patterns to minimize the need for device upgrades.

The following question focuses on these considerations for sustainability:

SUS 3: How do you take advantage of software and architecture patterns to support your sustainability goals?

Implement patterns for performing load smoothing and maintaining consistent high utilizati on of deployed resources to minimize the resources consumed. Components might become idle from lack of use because of changes in user behavior over time. Revise patterns and architect ure to consolidate under-utilized components to increase overall utilization. Retire component s that are no longer required. Understand the performance of your workload components, and optimize the components that consume the most resources. Be aware of the devices that your customers use to access your services, and implement patterns to minimize the need for device upgrades.

Optimize software and architecture for asynchronous and scheduled jobs: Use efficient software designs and architectures to minimize the average resources required per unit of work. Implement mechanisms that result in even utilization of components to reduce resources that are idle between tasks and minimize the impact of load spikes.

Remove or refactor workload components with low or no use: Monitor workload activity to identify changes in utilization of individual components over time. Remove components that are unused and no longer required, and refactor components with little utilization, to limit wasted resources.

Optimize areas of code that consume the most time or resources: Monitor workload activity to identify application components that consume the most resources. Optimize the code that runs within these components to minimize resource usage while maximizing performance.

Optimize impact on customer devices and equipment: Understand the devices and equipment that your customers use to consume your services, their expected lifecycle, and the financial and sustainability impact of replacing those components. Implement software patterns and architectures to minimize the need for customers to replace devices and upgrade equipment. For example, implement new features using code that is backward compatible with earlier hardware

and operating system versions, or manage the size of payloads so they don't exceed the storage capacity of the target device.

Use software patterns and architectures that most effectively supports data access and storage patterns: Understand how data is used within your workload, consumed by your users, transferred, and stored. Select technologies to minimize data processing and storage requirements.

Data management

The following question focuses on these considerations for sustainability:

SUS 4: How do you take advantage of data management policies and patterns to support your sustainability goals?

Implement data management practices to reduce the provisioned storage required to support your workload, and the resources required to use it. Understand your data, and use storage technologies and configurations that most effectively supports the business value of the data and how it's used. Lifecycle data to more efficient, less performant storage when requirements decrease, and delete data that's no longer required.

Implement a data classification policy: Classify data to understand its significance to business outcomes. Use this information to determine when you can move data to more energy-efficient storage or safely delete it.

Use technologies that support data access and storage patterns: Use storage that most effectively supports how your data is accessed and stored to minimize the resources provisioned while supporting your workload. For example, solid state devices (SSDs) are more energy intensive than magnetic drives and should be used only for active data use cases. Use energy-efficient, archival-class storage for infrequently accessed data.

Use lifecycle policies to delete unnecessary data: Manage the lifecycle of all your data and automatically enforce deletion timelines to minimize the total storage requirements of your workload.

Minimize over-provisioning in block storage: To minimize total provisioned storage, create block storage with size allocations that are appropriate for the workload. Use elastic volumes to expand storage as data grows without having to resize storage attached to compute resources. Regularly review elastic volumes and shrink over-provisioned volumes to fit the current data size.

Remove unneeded or redundant data: Duplicate data only when necessary to minimize total storage consumed. Use backup technologies that deduplicate data at the file and block level. Limit the use of Redundant Array of Independent Drives (RAID) configurations except where required to meet SLAs.

Use shared file systems or object storage to access common data: Adopt shared storage and single sources of truth to avoid data duplication and reduce the total storage requirements of your workload. Fetch data from shared storage only as needed. Detach unused volumes to release resources. Minimize data movement across networks: Use shared storage and access data from Regional data stores to minimize the total networking resources required to support data movement for your workload.

Back up data only when difficult to recreate: To minimize storage consumption, only back up data that has business value or is required to satisfy compliance requirements. Examine backup policies and exclude ephemeral storage that doesn't provide value in a recovery scenario.

Hardware and services

Look for opportunities to reduce workload sustainability impacts by making changes to your hardware management practices. Minimize the amount of hardware needed to provision and deploy, and select the most efficient hardware and services for your individual workload.

The following question focuses on these considerations for sustainability:

SUS 5: How do you select and use cloud hardware and services in your architecture to support your sustainability goals?

Look for opportunities to reduce workload sustainability impacts by making changes to your hardware management practices. Minimize the amount of hardware needed to provision and deploy, and select the most efficient hardware and services for your individual workload.

Use the minimum amount of hardware to meet your needs: Using the capabilities of the cloud, you can make frequent changes to your workload implementations. Update deployed components as your needs change.

Use instance types with the least impact: Continually monitor the release of new instance types and take advantage of energy efficiency improvements, including those instance types designed to support specific workloads such as machine learning training and inference, and video transcoding.

Use managed services: Managed services shift responsibility for maintaining high average utilization, and sustainability optimization of the deployed hardware, to AWS. Use managed services to distribute the sustainability impact of the service across all tenants of the service, reducing your individual contribution.

Optimize your use of GPUs: Graphics processing units (GPUs) can be a source of high-power consumption, and many GPU workloads are highly variable, such as rendering, transcoding, and machine learning training and modeling. Only run GPUs instances for the time needed, and decommission them with automation when not required to minimize resources consumed.

Process and culture

Look for opportunities to reduce your sustainability impact by making changes to your development, test, and deployment practices.

The following question focuses on these considerations for sustainability:

SUS 6: How do your organizational processes support your sustainability goals?

Look for opportunities to reduce your sustainability impact by making changes to your development, test, and deployment practices.

Adopt operations that can rapidly introduce sustainability improvements: Test and validate potential improvements before deploying them to production. Account for the cost of testing when calculating potential future benefit of an improvement. Develop low-cost testing operations to drive delivery of small improvements.

Keep your workload up to date: Up-to-date operating systems, libraries, and applications can improve workload efficiency and create adoption of more efficient technologies. Up-to-date software might also include features to measure the sustainability impact of your workload more accurately, as vendors deliver features to meet their own sustainability goals.

Increase utilization of build environments: Use automation and infrastructure as code to bring up pre-production environments when needed and take them down when not used. A common pattern is to schedule periods of availability that coincide with the working hours of your development team members. Hibernation is a useful tool to preserve state and rapidly bring instances online only when needed. Use instance types with burst capacity, Spot Instances, elastic database services, containers, and other technologies to align development and test capacity with use.

Use managed device farms for testing: Managed device farms spread the sustainability impact of hardware manufacturing and resource usage across multiple tenants. Managed device farms offer diverse device types so you can support earlier, less popular hardware, and avoid customer sustainability impact from unnecessary device upgrades.

Resources

Refer to the following resources to learn more about our best practices for sustainability.

Whitepaper

- Sustainability Pillar

Video

- The Climate Pledge

The review process

The review of architectures must be done in a consistent manner, with a blame-free approach that encourages diving deep. It should be a lightweight process (hours not days) that is a conversation and not an audit. The purpose of reviewing an architecture is to identify any critical issues that might need addressing or areas that could be improved. The outcome of the review is a set of actions that should improve the experience of a customer using the workload.

As discussed in the “On Architecture” section, you will want each team member to take responsibility for the quality of its architecture. We recommend that the team members who build an architecture use the Well-Architected Framework to continually review their architecture, rather than holding a formal review meeting. A nearly continuous approach permits your team members to update answers as the architecture evolves, and improve the architecture as you deliver features.

The AWS Well-Architected Framework is aligned to the way that AWS reviews systems and services internally. It is premised on a set of design principles that influences architectural approach, and questions that verify that people don’t neglect areas that often featured in Root Cause Analysis (RCA). Whenever there is a significant issue with an internal system, AWS service, or customer, we look at the RCA to see if we could improve the review processes we use.

Reviews should be applied at key milestones in the product lifecycle, early on in the design phase to avoid *one-way doors* that are difficult to change, and then before the go-live date. (Many decisions are reversible, two-way doors. Those decisions can use a lightweight process. One-way doors are hard or impossible to reverse and require more inspection before making them.) After you go into production, your workload will continue to evolve as you add new features and change technology implementations. The architecture of a workload changes over time. You must follow good hygiene practices to stop its architectural characteristics from degrading as you evolve it. As you make significant architecture changes, you should follow a set of hygiene processes including a Well-Architected review.

If you want to use the review as a one-time snapshot or independent measurement, you will want to verify that you have all the right people in the conversation. Often, we find that reviews are the first time that a team truly understands what they have implemented. An approach that works well when reviewing another team's workload is to have a series of informal conversations about their architecture where you can glean the answers to most questions. You can then follow up with one or two meetings where you can gain clarity or dive deep on areas of ambiguity or perceived risk.

Here are some suggested items to facilitate your meetings:

- A meeting room with whiteboards
- Print outs of any diagrams or design notes
- Action list of questions that require out-of-band research to answer (for example, “did we activate encryption or not?”)

After you have done a review, you should have a list of issues that you can prioritize based on your business context. You will also want to take into account the impact of those issues on the day-to-day work of your team. If you address these issues early, you could free up time to work on creating business value rather than solving recurring problems. As you address issues, you can update your review to see how the architecture is improving.

While the value of a review is clear after you have done one, you may find that a new team might be resistant at first. Here are some objections that can be handled through educating the team on the benefits of a review:

- “We are too busy!” (Often said when the team is getting ready for a significant launch.)
  - If you are getting ready for a big launch, you will want it to go smoothly. The review will permit you to understand any problems you might have missed.
  - We recommend that you carry out reviews early in the product lifecycle to uncover risks and develop a mitigation plan aligned with the feature delivery roadmap.
- “We don’t have time to do anything with the results!” (Often said when there is an immovable event, such as the Super Bowl, that they are targeting.)
  - These events can’t be moved. Do you really want to go into it without knowing the risks in your architecture? Even if you don’t address all of these issues you can still have playbooks for handling them if they materialize.
- “We don’t want others to know the secrets of our solution implementation!”
  - If you point the team at the questions in the Well-Architected Framework, they will see that none of the questions reveal any commercial or technical proprietary information.

As you carry out multiple reviews with teams in your organization, you might identify thematic issues. For example, you might see that a group of teams has clusters of issues in a particular pillar or topic. You will want to look at all your reviews in a holistic manner, and identify any mechanisms, training, or principal engineering talks that could help address those thematic issues.

Conclusion

The AWS Well-Architected Framework provides architectural best practices across the six pillars for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems in the cloud. The Framework provides a set of questions that allows you to review an existing or proposed architecture. It also provides a set of AWS best practices for each pillar. Using the Framework in your architecture will help you produce stable and efficient systems, which allow you to focus on your functional requirements.

Contributors

The following individuals and organizations contributed to this document:

- Brian Carlson, Operations Lead Well-Architected, Amazon Web Services
- Ben Potter, Security Lead Well-Architected, Amazon Web Services
- Seth Eliot, Reliability Lead Well-Architected, Amazon Web Services
- Eric Pullen, Sr. Solutions Architect, Amazon Web Services
- Rodney Lester, Principal Solutions Architect, Amazon Web Services
- Jon Steele, Sr. Technical Account Manager, Amazon Web Services
- Max Ramsay, Principal Security Solutions Architect, Amazon Web Services
- Callum Hughes, Solutions Architect, Amazon Web Services
- Ben Mergen, Senior Cost Lead Solutions Architect, Amazon Web Services
- Chris Kozlowski, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Alex Livingstone, Principal Specialist Solutions Architect, Cloud Operations, Amazon Web Services
- Paul Moran, Principal Technologist, Enterprise Support, Amazon Web Services
- Peter Mullen, Advisory Consultant, Professional Services, Amazon Web Services
- Chris Pates, Senior Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Arvind Raghunathan, Principal Specialist Technical Account Manager, Enterprise Support, Amazon Web Services
- Sam Mokhtari, Senior Efficiency Lead Solutions Architect, Amazon Web Services

Further reading

AWS Architecture Center

AWS Cloud Compliance

AWS Well-Architected Partner program

AWS Well-Architected Tool

AWS Well-Architected homepage

Operational Excellence Pillar whitepaper

Security Pillar whitepaper

Reliability Pillar whitepaper

Performance Efficiency Pillar whitepaper

Cost Optimization Pillar whitepaper

Sustainability Pillar whitepaper

The Amazon Builders' Library

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Major update	Best practices were updated with new guidance in Reliability, Security, Operational excellence, Sustainability, and Performance efficiency. The Reliability pillar received a large-scale refresh and update to many best practices. Guidance across Security and Operational excellence has been updated and refined with new service and generative AI suggestions. Sustainability received multiple updates based on AWS services and a new best practice.	November 6, 2024
Major update	Large-scale best practice updates were made throughout the pillars. Security and cost both received new best practices.	June 27, 2024
Major update	Major pillar updates.	October 3, 2023
Major update	Best practices updated with prescriptive guidance and new best practices added. New questions added to the	April 10, 2023

	Security and Cost Optimization pillars.	
Minor update	Added definition for level of effort and updated best practices in the appendix.	October 20, 2022
Major update	Added Sustainability Pillar and updated links.	December 2, 2021
Major update	Sustainability Pillar added to the framework.	November 20, 2021
Minor update	Removed non-inclusive language.	April 22, 2021
Minor update	Fixed numerous links.	March 10, 2021
Minor update	Minor editorial changes throughout.	July 15, 2020
Major update	Review and rewrite of most questions and answers.	July 8, 2020
Whitepaper updated	Addition of AWS Well-Architected Tool, links to AWS Well-Architected Labs, and AWS Well-Architected Partners, minor fixes to enable multiple language version of framework.	July 1, 2019

Whitepaper updated	Review and rewrite of most questions and answers, to ensure questions focus on one topic at a time. This caused some previous questions to be split into multiple questions. Added common terms to definitions (workload , component etc). Changed presentation of question in main body to include descripti ve text.	November 1, 2018
Whitepaper updated	Updates to simplify question text, standardize answers, and improve readability.	June 1, 2018
Whitepaper updated	Operational Excellence moved to front of pillars and rewritten so it frames other pillars. Refreshed other pillars to reflect evolution of AWS.	November 1, 2017
Whitepaper updated	Updated the Framework to include operational excellenc e pillar, and revised and updated the other pillars to reduce duplication and incorporate learnings from carrying out reviews with thousands of customers.	November 1, 2016
Minor updates	Updated the Appendix with current Amazon CloudWatch Logs information.	November 1, 2015

Initial publication	AWS Well-Architected Framework published.	October 1, 2015

Note

To subscribe to RSS updates, you must have an RSS plugin enabled for the browser that you are using.

Framework versions:

- 2024-06-27
- 2023-10-03
- 2023-04-10
- 2022-03-31

Appendix: Questions and best practices

This appendix summarizes all the questions and best practices in the AWS Well-Architected Framework.

Pillars

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Sustainability

Operational excellence

Operational excellence (OE) is a commitment to build software correctly while consistently delivering a great customer experience. The operational excellence pillar contains best practices for organizing your team, designing your workload, operating it at scale, and evolving it over time. You can find prescriptive guidance on implementation in the Operational Excellence Pillar whitepaper.

Best practice areas

- Organization
- Prepare
- Operate
- Evolve

Organization

Questions

- OPS 1. How do you determine what your priorities are?
- OPS 2. How do you structure your organization to support your business outcomes?
- OPS 3. How does your organizational culture support your business outcomes?

OPS 1. How do you determine what your priorities are?

Everyone should understand their part in enabling business success. Have shared goals in order to set priorities for resources. This will maximize the benefits of your efforts.

Best practices

- OPS01-BP01 Evaluate external customer needs
- OPS01-BP02 Evaluate internal customer needs
- OPS01-BP03 Evaluate governance requirements
- OPS01-BP04 Evaluate compliance requirements
- OPS01-BP05 Evaluate threat landscape
- OPS01-BP06 Evaluate tradeoffs while managing benefits and risks

OPS01-BP01 Evaluate external customer needs

Involve key stakeholders, including business, development, and operations teams, to determine where to focus efforts on external customer needs. This verifies that you have a thorough understanding of the operations support that is required to achieve your desired business outcomes.

Desired outcome:

- You work backwards from customer outcomes.
- You understand how your operational practices support business outcomes and objectives.
- You engage all relevant parties.
- You have mechanisms to capture external customer needs.

Common anti-patterns:

- You have decided not to have customer support outside of core business hours, but you haven't reviewed historical support request data. You do not know whether this will have an impact on your customers.
- You are developing a new feature but have not engaged your customers to find out if it is desired, if desired in what form, and without experimentation to validate the need and method of delivery.

Benefits of establishing this best practice: Customers whose needs are satisfied are much more likely to remain customers. Evaluating and understanding external customer needs will inform how you prioritize your efforts to deliver business value.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understand business needs: Business success is created by shared goals and understanding across stakeholders, including business, development, and operations teams.

Review business goals, needs, and priorities of external customers: Engage key stakeholders, including business, development, and operations teams, to discuss goals, needs, and priorities of external customers. This ensures that you have a thorough understanding of the operational support that is required to achieve business and customer outcomes.

Establish a shared understanding: Establish a shared understanding of the business functions of the workload, the roles of each of the teams in operating the workload, and how these factors support your shared business goals across internal and external customers.

Resources

Related best practices:

- OPS11-BP03 Implement feedback loops

OPS01-BP02 Evaluate internal customer needs

Involve key stakeholders, including business, development, and operations teams, when determining where to focus efforts on internal customer needs. This will ensure that you have a thorough understanding of the operations support that is required to achieve business outcomes.

Desired outcome:

- You use your established priorities to focus your improvement efforts where they will have the greatest impact (for example, developing team skills, improving workload performance, reducing costs, automating runbooks, or enhancing monitoring).
- You update your priorities as needs change.

Common anti-patterns:

- You have decided to change IP address allocations for your product teams, without consulting them, to make managing your network easier. You do not know the impact this will have on your product teams.
- You are implementing a new development tool but have not engaged your internal customers to find out if it is needed or if it is compatible with their existing practices.
- You are implementing a new monitoring system but have not contacted your internal customers to find out if they have monitoring or reporting needs that should be considered.

Benefits of establishing this best practice: Evaluating and understanding internal customer needs informs how you prioritize your efforts to deliver business value.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Understand business needs: Business success is created by shared goals and understanding across stakeholders including business, development, and operations teams.
- Review business goals, needs, and priorities of internal customers: Engage key stakeholders, including business, development, and operations teams, to discuss goals, needs, and priorities of internal customers. This ensures that you have a thorough understanding of the operational support that is required to achieve business and customer outcomes.
- Establish shared understanding: Establish shared understanding of the business functions of the workload, the roles of each of the teams in operating the workload, and how these factors support shared business goals across internal and external customers.

Resources

Related best practices:

- OPS11-BP03 Implement feedback loops

OPS01-BP03 Evaluate governance requirements

Governance is the set of policies, rules, or frameworks that a company uses to achieve its business goals. Governance requirements are generated from within your organization. They can affect the types of technologies you choose or influence the way you operate your workload. Incorporate

organizational governance requirements into your workload. Conformance is the ability to demonstrate that you have implemented governance requirements.

Desired outcome:

- Governance requirements are incorporated into the architectural design and operation of your workload.
- You can provide proof that you have followed governance requirements.
- Governance requirements are regularly reviewed and updated.

Common anti-patterns:

- Your organization mandates that the root account has multi-factor authentication. You failed to implement this requirement and the root account is compromised.
- During the design of your workload, you choose an instance type that is not approved by the IT department. You are unable to launch your workload and must conduct a redesign.
- You are required to have a disaster recovery plan. You did not create one and your workload suffers an extended outage.
- Your team wants to use new instances but your governance requirements have not been updated to allow them.

Benefits of establishing this best practice:

- Following governance requirements aligns your workload with larger organization policies.
- Governance requirements reflect industry standards and best practices for your organization.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify governance requirement by working with stakeholders and governance organizations. Include governance requirements into your workload. Be able to demonstrate proof that you've followed governance requirements.

Customer example

At AnyCompany Retail, the cloud operations team works with stakeholders across the organization to develop governance requirements. For example, they prohibit SSH access into Amazon EC2

instances. If teams need system access, they are required to use AWS Systems Manager Session Manager. The cloud operations team regularly updates governance requirements as new services become available.

Implementation steps

1. Identify the stakeholders for your workload, including any centralized teams.
2. Work with stakeholders to identify governance requirements.
3. Once you've generated a list, prioritize the improvement items, and begin implementing them into your workload.
   a. Use services like AWS Config to create governance-as-code and validate that governance requirements are followed.
   b. If you use AWS Organizations, you can leverage Service Control Policies to implement governance requirements.
4. Provide documentation that validates the implementation.

Level of effort for the implementation plan: Medium. Implementing missing governance requirements may result in rework of your workload.

Resources

Related best practices:

- OPS01-BP04 Evaluate compliance requirements - Compliance is like governance but comes from outside an organization.

Related documents:

- AWS Management and Governance Cloud Environment Guide
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- Governance in the AWS Cloud: The Right Balance Between Agility and Safety
- What is Governance, Risk, And Compliance (GRC)?

Related videos:

- AWS Management and Governance: Configuration, Compliance, and Audit - AWS Online Tech Talks

- AWS re:Inforce 2019: Governance for the Cloud Age (DEM12-R1)
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- AWS re:Invent 2020: Agile governance on AWS GovCloud (US)

Related examples:

- AWS Config Conformance Pack Samples

Related services:

- AWS Config
- AWS Organizations - Service Control Policies

OPS01-BP04 Evaluate compliance requirements

Regulatory, industry, and internal compliance requirements are an important driver for defining your organization's priorities. Your compliance framework may preclude you from using specific technologies or geographic locations. Apply due diligence if no external compliance frameworks are identified. Generate audits or reports that validate compliance.

If you advertise that your product meets specific compliance standards, you must have an internal process for ensuring continuous compliance. Examples of compliance standards include PCI DSS, FedRAMP, and HIPAA. Applicable compliance standards are determined by various factors, such as what types of data the solution stores or transmits and which geographic regions the solution supports.

Desired outcome:

- Regulatory, industry, and internal compliance requirements are incorporated into architectural selection.
- You can validate compliance and generate audit reports.

Common anti-patterns:

- Parts of your workload fall under the Payment Card Industry Data Security Standard (PCI-DSS) framework but your workload stores credit cards data unencrypted.

- Your software developers and architects are unaware of the compliance framework that your organization must adhere to.
- The yearly Systems and Organizations Control (SOC2) Type II audit is happening soon and you are unable to verify that controls are in place.

Benefits of establishing this best practice:

- Evaluating and understanding the compliance requirements that apply to your workload will inform how you prioritize your efforts to deliver business value.
- You choose the right locations and technologies that are congruent with your compliance framework.
- Designing your workload for auditability helps you to prove you are adhering to your compliance framework.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing this best practice means that you incorporate compliance requirements into your architecture design process. Your team members are aware of the required compliance framework. You validate compliance in line with the framework.

Customer example

AnyCompany Retail stores credit card information for customers. Developers on the card storage team understand that they need to comply with the PCI-DSS framework. They've taken steps to verify that credit card information is stored and accessed securely in line with the PCI-DSS framework. Every year they work with their security team to validate compliance.

Implementation steps

1. Work with your security and governance teams to determine what industry, regulatory, or internal compliance frameworks that your workload must adhere to. Incorporate the compliance frameworks into your workload.
   a. Validate continual compliance of AWS resources with services like AWS Compute Optimizer and AWS Security Hub CSPM.

2. Educate your team members on the compliance requirements so they can operate and evolve the workload in line with them. Compliance requirements should be included in architectural and technological choices.
3. Depending on the compliance framework, you may be required to generate an audit or compliance report. Work with your organization to automate this process as much as possible.
   a. Use services like AWS Audit Manager to validate compliance and generate audit reports.
   b. You can download AWS security and compliance documents with AWS Artifact.

Level of effort for the implementation plan: Medium. Implementing compliance frameworks can be challenging. Generating audit reports or compliance documents adds additional complexity.

Resources

Related best practices:

- SEC01-BP03 Identify and validate control objectives - Security control objectives are an important part of overall compliance.
- SEC01-BP06 Automate testing and validation of security controls in pipelines - As part of your pipelines, validate security controls. You can also generate compliance documentation for new changes.
- SEC07-BP02 Define data protection controls - Many compliance frameworks have data handling and storage policies based.
- SEC10-BP03 Prepare forensic capabilities - Forensic capabilities can sometimes be used in auditing compliance.

Related documents:

- AWS Compliance Center
- AWS Compliance Resources
- AWS Risk and Compliance Whitepaper
- AWS Shared Responsibility Model
- AWS services in scope by compliance programs

Related videos:

- AWS re:Invent 2020: Achieve compliance as code using AWS Compute Optimizer
- AWS re:Invent 2021 - Cloud compliance, assurance, and auditing
- AWS Summit ATL 2022 - Implementing compliance, assurance, and auditing on AWS (COP202)

Related examples:

- PCI DSS and AWS Foundational Security Best Practices on AWS

Related services:

- AWS Artifact
- AWS Audit Manager
- AWS Compute Optimizer
- AWS Security Hub CSPM

OPS01-BP05 Evaluate threat landscape

Evaluate threats to the business (for example, competition, business risk and liabilities, operational risks, and information security threats) and maintain current information in a risk registry. Include the impact of risks when determining where to focus efforts.

The Well-Architected Framework emphasizes learning, measuring, and improving. It provides a consistent approach for you to evaluate architectures, and implement designs that will scale over time. AWS provides the AWS Well-Architected Tool to help you review your approach prior to development, the state of your workloads prior to production, and the state of your workloads in production. You can compare them to the latest AWS architectural best practices, monitor the overall status of your workloads, and gain insight to potential risks.

AWS customers are eligible for a guided Well-Architected Review of their mission-critical workloads to measure their architectures against AWS best practices. Enterprise Support customers are eligible for an Operations Review, designed to help them to identify gaps in their approach to operating in the cloud.

The cross-team engagement of these reviews helps to establish common understanding of your workloads and how team roles contribute to success. The needs identified through the review can help shape your priorities.

AWS Trusted Advisor is a tool that provides access to a core set of checks that recommend optimizations that may help shape your priorities. Business and Enterprise Support customers receive access to additional checks focusing on security, reliability, performance, and cost-optimization that can further help shape their priorities.

Desired outcome:

- You regularly review and act on Well-Architected and Trusted Advisor outputs
- You are aware of the latest patch status of your services
- You understand the risk and impact of known threats and act accordingly
- You implement mitigations as necessary
- You communicate actions and context

Common anti-patterns:

- You are using an old version of a software library in your product. You are unaware of security updates to the library for issues that may have unintended impact on your workload.
- Your competitor just released a version of their product that addresses many of your customers' complaints about your product. You have not prioritized addressing any of these known issues.
- Regulators have been pursuing companies like yours that are not compliant with legal regulatory compliance requirements. You have not prioritized addressing any of your outstanding compliance requirements.

Benefits of establishing this best practice: You identify and understand the threats to your organization and workload, which helps your determination of which threats to address, their priority, and the resources necessary to do so.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- **Evaluate threat landscape:** Evaluate threats to the business (for example, competition, business risk and liabilities, operational risks, and information security threats), so that you can include their impact when determining where to focus efforts.
  - AWS Latest Security Bulletins
  - AWS Trusted Advisor

- **Maintain a threat model:** Establish and maintain a threat model identifying potential threats, planned and in place mitigations, and their priority. Review the probability of threats manifesting as incidents, the cost to recover from those incidents and the expected harm caused, and the cost to prevent those incidents. Revise priorities as the contents of the threat model change.

Resources

Related best practice:

- SEC01-BP07 Identify threats and prioritize mitigations using a threat model

Related documents:

- AWS Cloud Compliance
- AWS Latest Security Bulletins
- AWS Trusted Advisor

Related videos:

- AWS re:Inforce 2023 - A tool to help improve your threat modeling

OPS01-BP06 Evaluate tradeoffs while managing benefits and risks

Competing interests from multiple parties can make it challenging to prioritize efforts, build capabilities, and deliver outcomes aligned with business strategies. For example, you may be asked to accelerate speed-to-market for new features over optimizing IT infrastructure costs. This can put two interested parties in conflict with one another. In these situations, decisions need to be brought to a higher authority to resolve conflict. Data is required to remove emotional attachment from the decision-making process.

The same challenge may occur at a tactical level. For example, the choice between using relational or non-relational database technologies can have a significant impact on the operation of an application. It's critical to understand the predictable results of various choices.

AWS can help you educate your teams about AWS and its services to increase their understanding of how their choices can have an impact on your workload. Use the resources provided by Support (AWS Knowledge Center, AWS Discussion Forums, and Support Center) and AWS Documentation to educate your teams. For further questions, reach out to Support.

AWS also shares operational best practices and patterns in The Amazon Builders' Library. A wide variety of other useful information is available through the AWS Blog and The Official AWS Podcast.

Desired outcome: You have a clearly defined decision-making governance framework to facilitate important decisions at every level within your cloud delivery organization. This framework includes features like a risk register, defined roles that are authorized to make decisions, and a defined models for each level of decision that can be made. This framework defines in advance how conflicts are resolved, what data needs to be presented, and how options are prioritized, so that once decisions are made you can commit without delay. The decision-making framework includes a standardized approach to reviewing and weighing the benefits and risks of every decision to understand the tradeoffs. This may include external factors, such as adherence to regulatory compliance requirements.

Common anti-patterns:

- Your investors request that you demonstrate compliance with Payment Card Industry Data Security Standards (PCI DSS). You do not consider the tradeoffs between satisfying their request and continuing with your current development efforts. Instead, you proceed with your development efforts without demonstrating compliance. Your investors stop their support of your company over concerns about the security of your platform and their investments.
- You have decided to include a library that one of your developers found on the internet. You have not evaluated the risks of adopting this library from an unknown source and do not know if it contains vulnerabilities or malicious code.
- The original business justification for your migration was based upon the modernization of 60% of your application workloads. However, due to technical difficulties, a decision was made to modernize only 20%, leading to a reduction in planned benefits long-term, increased operator toil for infrastructure teams to manually support legacy systems, and greater reliance on developing new skillsets in your infrastructure teams that were not planning for this change.

Benefits of establishing this best practice: Fully aligning and supporting board-level business priorities, understanding the risks to achieving success, making informed decisions, and acting appropriately when risks impede chances for success. Understanding the implications and consequences of your decisions helps you to prioritize your options and bring leaders into agreement faster, leading to improved business outcomes. Identifying the available benefits of your choices and being aware of the risks to your organization helps you make data-driven decisions, rather than relying on anecdotes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Managing benefits and risks should be defined by a governing body that drives the requirements for key decision-making. You want decisions to be made and prioritized based on how they benefit the organization, with an understanding of the risks involved. Accurate information is critical for making the organizational decisions. This should be based on solid measurements and defined by common industry practices of cost benefit analysis. To make these types of decisions, strike a balance between centralized and decentralized authority. There is always a tradeoff, and it's important to understand how each choice impacts defined strategies and desired business outcomes.

Implementation steps

1. Formalize benefits measurement practices within a holistic cloud governance framework.
    a. Balance central control of decision-making with decentralized authority for some decisions.
    b. Understand that burdensome decision-making processes imposed on every decision can slow you down.
    c. Incorporate external factors into your decision making process (like compliance requirements).
2. Establish an agreed-upon decision-making framework for various levels of decisions, which includes who is required to unblock decisions that are subject to conflicted interests.
    a. Centralize one-way door decisions that could be irreversible.
    b. Allow two-way door decisions to be made by lower level organizational leaders.
3. Understand and manage benefits and risks. Balance the benefits of decisions against the risks involved.
    a. **Identify benefits**: Identify benefits based on business goals, needs, and priorities. Examples include business case impact, time-to-market, security, reliability, performance, and cost.
    b. **Identify risks**: Identify risks based on business goals, needs, and priorities. Examples include time-to-market, security, reliability, performance, and cost.
    c. **Assess benefits against risks and make informed decisions**: Determine the impact of benefits and risks based on goals, needs, and priorities of your key stakeholders, including business, development, and operations. Evaluate the value of the benefit against the probability of the risk being realized and the cost of its impact. For example, emphasizing speed-to-market over reliability might provide competitive advantage. However, it may result in reduced uptime if there are reliability issues.

4. Programatically enforce key decisions that automate your adherence to compliance requirements.
5. Leverage known industry frameworks and capabilities, such as Value Stream Analysis and LEAN, to baseline current state performance, business metrics, and define iterations of progress towards improvements to these metrics.

Level of effort for the implementation plan: Medium-High

Resources

Related best practices:

- OPS01-BP05 Evaluate threat landscape

Related documents:

- Elements of Amazon's Day 1 Culture | Make high quality, high velocity decisions
- Cloud Governance
- Management & Governance Cloud Environment
- Governance in the Cloud and in the Digital Age: Parts One & Two

Related videos:

- Podcast | Jeff Bezos | On how to make decisions

Related examples:

- Make informed decisions using data (The DevOps Sagas)
- Using development value stream mapping to identify constraints to DevOps outcomes

OPS 2. How do you structure your organization to support your business outcomes?

Your teams must understand their part in achieving business outcomes. Teams should understand their roles in the success of other teams, the role of other teams in their success, and have shared

goals. Understanding responsibility, ownership, how decisions are made, and who has authority to make decisions will help focus efforts and maximize the benefits from your teams.

Best practices

- OPS02-BP01 Resources have identified owners
- OPS02-BP02 Processes and procedures have identified owners
- OPS02-BP03 Operations activities have identified owners responsible for their performance
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership
- OPS02-BP05 Mechanisms exist to request additions, changes, and exceptions
- OPS02-BP06 Responsibilities between teams are predefined or negotiated

OPS02-BP01 Resources have identified owners

Resources for your workload must have identified owners for change control, troubleshooting, and other functions. Owners are assigned for workloads, accounts, infrastructure, platforms, and applications. Ownership is recorded using tools like a central register or metadata attached to resources. The business value of components informs the processes and procedures applied to them.

Desired outcome:

- Resources have identified owners using metadata or a central register.
- Team members can identify who owns resources.
- Accounts have a single owner where possible.

Common anti-patterns:

- The alternate contacts for your AWS accounts are not populated.
- Resources lack tags that identify what teams own them.
- You have an ITSM queue without an email mapping.
- Two teams have overlapping ownership of a critical piece of infrastructure.

Benefits of establishing this best practice:

- Change control for resources is straightforward with assigned ownership.

- You can involve the right owners when troubleshooting issues.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Define what ownership means for the resource use cases in your environment. Ownership can mean who oversees changes to the resource, supports the resource during troubleshooting, or who is financially accountable. Specify and record owners for resources, including name, contact information, organization, and team.

Customer example

AnyCompany Retail defines ownership as the team or individual that owns changes and support for resources. They leverage AWS Organizations to manage their AWS accounts. Alternate account contacts are configuring using group inboxes. Each ITSM queue maps to an email alias. Tags identify who own AWS resources. For other platforms and infrastructure, they have a wiki page that identifies ownership and contact information.

Implementation steps

1. Start by defining ownership for your organization. Ownership can imply who owns the risk for the resource, who owns changes to the resource, or who supports the resource when troubleshooting. Ownership could also imply financial or administrative ownership of the resource.
2. Use AWS Organizations to manage accounts. You can manage the alternate contacts for your accounts centrally.
   a. Using company owned email addresses and phone numbers for contact information helps you to access them even if the individuals whom they belong to are no longer with your organization. For example, create separate email distribution lists for billing, operations, and security and configure these as Billing, Security, and Operations contacts in each active AWS account. Multiple people will receive AWS notifications and be able to respond, even if someone is on vacation, changes roles, or leaves the company.
   b. If an account is not managed by AWS Organizations, alternate account contacts help AWS get in contact with the appropriate personnel if needed. Configure the account's alternate contacts to point to a group rather than an individual.
3. Use tags to identify owners for AWS resources. You can specify both owners and their contact information in separate tags.

   a. You can use AWS Config rules to enforce that resources have the required ownership tags.
   b. For in-depth guidance on how to build a tagging strategy for your organization, see AWS Tagging Best Practices whitepaper.
4. Use Amazon Q Business, a conversational assistant that uses generative AI to enhance workforce productivity, answer questions, and complete tasks based on information in your enterprise systems.
   a. Connect Amazon Q Business to your company's data source. Amazon Q Business offers prebuilt connectors to over 40 supported data sources, including Amazon Simple Storage Service (Amazon S3), Microsoft SharePoint, Salesforce, and Atlassian Confluence. For more information, see Amazon Q Business connectors.
5. For other resources, platforms, and infrastructure, create documentation that identifies ownership. This should be accessible to all team members.

Level of effort for the implementation plan: Low. Leverage account contact information and tags to assign ownership of AWS resources. For other resources you can use something as simple as a table in a wiki to record ownership and contact information, or use an ITSM tool to map ownership.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership

Related documents:

- AWS Account Management - Updating contact information
- AWS Organizations - Updating alternative contacts in your organization
- AWS Tagging Best Practices whitepaper
- Build private and secure enterprise generative AI apps with Amazon Q Business and AWS IAM Identity Center
- Amazon Q Business, now generally available, helps boost workforce productivity with generative AI
- AWS Cloud Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations

- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

Related workshops:

- AWS Workshop - Tagging

Related examples:

- AWS Config Rules - Amazon EC2 with required tags and valid values

Related services:

- AWS Config Rules - required-tags
- AWS Organizations

OPS02-BP02 Processes and procedures have identified owners

Understand who has ownership of the definition of individual processes and procedures, why those specific process and procedures are used, and why that ownership exists. Understanding the reasons that specific processes and procedures are used aids in identification of improvement opportunities.

Desired outcome: Your organization has a well defined and maintained set of process and procedures for operational tasks. The process and procedures are stored in a central location and available to your team members. Process and procedures are updated frequently, by clearly assigned ownership. Where possible, scripts, templates, and automation documents are implemented as code.

Common anti-patterns:

- Processes are not documented. Fragmented scripts may exist on isolated operator workstations.
- Knowledge of how to use scripts is held by a few individuals or informally as team knowledge.
- A legacy process is due for an update, but ownership of the update is unclear, and the original author is no longer part of the organization.
- Processes and scripts are not discoverable, so they are not readily available when required (for example, in response to an incident).

Benefits of establishing this best practice:

- Processes and procedures boost your efforts to operate your workloads.
- New team members become effective more quickly.
- Reduced time to mitigate incidents.
- Different team members (and teams) can use the same processes and procedures in a consistent manner.
- Teams can scale their processes with repeatable processes.
- Standardized processes and procedures help mitigate the impact of transferring workload responsibilities between teams.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Processes and procedures have identified owners who are responsible for their definition.
  - Identify the operations activities conducted in support of your workloads. Document these activities in a discoverable location.
  - Uniquely identify the individual or team responsible for the specification of an activity. They are responsible to verify that it can be successfully performed by an adequately skilled team member with the correct permissions, access, and tools. If there are issues with performing that activity, the team members performing it are responsible for providing the detailed feedback necessary for the activity to be improved.
  - Capture ownership in the metadata of the activity artifact through services like AWS Systems Manager, through documents, and AWS Lambda. Capture resource ownership using tags or resource groups, specifying ownership and contact information. Use AWS Organizations to create tagging polices and capture ownership and contact information.
- Over time, these procedures should be evolved to be runnable as code, reducing the need for human intervention.
  - For example, consider AWS Lambda functions, CloudFormation templates, or AWS Systems Manager automation docs.
  - Perform version control in appropriate repositories.
  - Include suitable resource tagging so owners and documentation can readily be identified.

Customer example

AnyCompany Retail defines ownership as the team or individual that owns processes for an application or groups of applications (that share common architetural practices and technologies). Initially, the process and procedures are documented as step-by-step guides in the document management system, discoverable using tags on the AWS account that hosts the application and on specific groups of resources within the account. They leverage AWS Organizations to manage their AWS accounts. Over time, these processes are converted to code, and resources are defined using infrastructure as code (such as CloudFormation or AWS Cloud Development Kit (AWS CDK) templates). The operational processes become automation documents in AWS Systems Manager or AWS Lambda functions, which can be initiated as scheduled tasks, in response to events such as AWS CloudWatch alarms or AWS EventBridge events, or started by requests within an IT service management (ITSM) platform. All process have tags to identify ownership. Documentation for the automation and process is maintained within the wiki pages generated by the code repository for the process.

Implementation steps

1. Document the existing processes and procedures.
    a. Review and keep them up-to-date.
    b. Identify an owner for each process or procedure.
    c. Place them under version control.
    d. Where possible, share processes and procedures across workloads and environments that share architectural designs.
2. Establish mechanisms for feedback and improvement.
    a. Define policies for how frequently processes should be reviewed.
    b. Define processes for reviewers and approvers.
    c. Implement issues or a ticketing queue for feedback to be provided and tracked.
    d. Whereever possible, processes and procedures should have pre-approval and risk classification from a change approval board (CAB).
3. Verify that processes and procedures are accessible and discoverable by those who need to run them.
    a. Use tags to indicate where the process and procedures can be accessed for the workload.
    b. Use meaningful error and event messaging to indicate the appropriate processes or procedures to address an issue.
    c. Use wikis and document management, and make processes and procedures searchable consistently accross the organization.

4. Use Amazon Q Business, a conversational assistant that uses generative AI to enhance workforce productivity, answer questions, and complete tasks based on information in your enterprise systems.
    a. Connect Amazon Q Business to your company's data source. Amazon Q Business offers prebuilt connectors to over 40 supported data sources, including Amazon S3, Microsoft SharePoint, Salesforce, and Atlassian Confluence. For more information, see Amazon Q connectors.
5. Automate when appropriate.
    a. Automations should be developed when services and technologies provide an API.
    b. Educate adequately on processes. Develop the user stories and requirements to automate those processes.
    c. Measure the use of your processes and procedures successfully, and create issues or tickets to support iterative improvement.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS02-BP01 Resources have identified owners
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership
- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Whitepaper - Introduction to DevOps on AWS
- AWS Whitepaper - Best Practices for Tagging AWS Resources
- AWS Whitepaper - Organizing Your AWS Environment Using Multiple Accounts
- AWS Cloud Operations and Migrations Blog - Using Amazon Q Business to streamline your operations
- AWS Cloud Operations & Migrations Blog - Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Cloud Operations & Migrations Blog - Implementing automated and centralized tagging controls with AWS Config and AWS Organizations

- AWS Security Blog - Extend your pre-commit hooks with AWS CloudFormation Guard
- AWS DevOps Blog - Integrating AWS CloudFormation Guard into CI/CD pipelines

Related workshops:

- AWS Well-Architected Operational Excellence Workshop
- AWS Workshop - Tagging

Related videos:

- How to automate IT Operations on AWS
- AWS re:Invent 2020 - Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 - Automating patch management and compliance using AWS (NIS306)
- Supports You - Diving Deep into AWS Systems Manager

Related services:

- AWS Systems Manager - Automation
- AWS Service Management Connector

OPS02-BP03 Operations activities have identified owners responsible for their performance

Understand who has responsibility to perform specific activities on defined workloads and why that responsibility exists. Understanding who has responsibility to perform activities informs who will conduct the activity, validate the result, and provide feedback to the owner of the activity.

Desired outcome:

Your organization clearly defines responsibilities to perform specific activities on defined workloads and respond to events generated by the workload. The organization documents ownership of processes and fulfillment and makes this information discoverable. You review and update responsibilities when organizational changes take place, and teams track and measure the performance of defect and inefficiency identification activities. You implement feedback mechanisms to track defects and improvements and support iterative improvement.

Common anti-patterns:

- You do not document responsibilities.
- Fragmented scripts exist on isolated operator workstations. Only a few individuals know how to use them or informally refer to them as *team knowledge*.
- A legacy process is due for update, but no one knows who owns the process, and the original author is no longer part of the organization.
- Processes and scripts can't be discovered, and they are not readily available when required (for example, in response to an incident).

Benefits of establishing this best practice:

- You understand who is responsible to perform an activity, who to notify when action is needed, and who performs the action, validates the result, and provides feedback to the owner of the activity.
- Processes and procedures boost your efforts to operate your workloads.
- New team members become effective more quickly.
- You reduce the time it takes to mitigate incidents.
- Different teams use the same processes and procedures to perform tasks in a consistent manner.
- Teams can scale their processes with repeatable processes.
- Standardized processes and procedures help mitigate the impact of transferring workload responsibilties between teams.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To begin to define responsibilities, start with existing documentation, like responsibility matrices, processes and procedures, roles and responsibilities, and tools and automation. Review and host discussions on the responsibilities for documented processes. Review with teams to identify misalignments between document responsibilities and processes. Discuss services offered with internal customers of that team to identify expectations gaps between teams.

Analyze and address the discrepancies. Identify opportunities to improvement, and look for frequently requested, resource-intensive activities, which are typically strong candidates for improvement. Explore best practices, patterns, and prescriptive guidance to simplify and standardize improvements. Record improvement opportunities, and track the improvements to completion.

Over time, these procedures should be evolved to be run as code, reducing the need for human intervention. For example, procedures can be initiated as AWS Lambda functions, CloudFormation templates, or AWS Systems Manager Automation documents. Verify that these procedures are version-controlled in appropriate repositories, and include suitable resource tagging so that teams can readily identify owners and documentation. Document the responsibility for carrying out the activities, and then monitor the automations for successful initiation and operation, as well as performance of the desired outcomes.

Customer example

AnyCompany Retail defines ownership as the team or individual that owns processes for an application or groups of applications that share common architectural practices and technologies. Initially, the company documents the processes and procedures as step-by-step guides in the document management system. They make the procedures discoverable using tags on the AWS account that hosts the application and on specific groups of resources within the account, using AWS Organizations to manage their AWS accounts. Over time, AnyCompany Retail converts these processes to code and defines resources using infrastructure as code (through services like CloudFormation or AWS Cloud Development Kit (AWS CDK) templates). The operational processes become Automation documents in AWS Systems Manager or AWS Lambda functions, which can be initiated as scheduled tasks in response to events such as Amazon CloudWatch alarms or Amazon EventBridge events or by requests within an IT service management (ITSM) platform. All processes have tags to identify who owns them. Teams manage documentation for the automation and process within the wiki pages generated by the code repository for the process.

Implementation steps

1. Document the existing processes and procedures.
    a. Review and verify that they are up-to-date.
    b. Verify that each process or procedure has an owner.
    c. Place the procedures under version control.
    d. Where possible, share processes and procedures across workloads and environments that share architectural designs.
2. Establish mechanisms for feedback and improvement.
    a. Define policies for how frequently processes should be reviewed.
    b. Define processes for reviewers and approvers.
    c. Implement issues or a ticketing queue to provide and track feedback.

   d. Wherever possible, provide pre-approval and risk classification for processes and procedures from a change approval board (CAB).
3. Make process and procedures accessible and discoverable by users who need to run them.
   a. Use tags to indicate where the process and procedures can be accessed for the workload.
   b. Use meaningful error and event messaging to indicate the appropriate process or proceedure to address the issue.
   c. Use wikis or document management to make processes and procedures consistently searchable across the organization.
4. Automate when it is appropriate to do so.
   a. Where services and technologies provide an API, develop automations.
   b. Verify that processes are well-understood, and develop the user stories and requirements to automate those processes.
   c. Measure the successful use of processes and procedures, with issue tracking to support iterative improvement.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS02-BP01 Resources have identified owners
- OPS02-BP02 Processes and procedures have identified owners
- OPS02-BP04 Mechanisms exist to manage responsibilities and ownership
- OPS02-BP05 Mechanisms exist to identify responsibility and ownership
- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Whitepaper | Introduction to DevOps on AWS
- AWS Whitepaper | Best Practices for Tagging AWS Resources
- AWS Whitepaper | Organizing Your AWS Environment Using Multiple Accounts
- AWS Cloud Operations & Migrations Blog | Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services

- AWS Workshop - Tagging
- AWS Service Management Connector

Related videos:

- AWS Knowledge Center Live | Tagging AWS Resources
- AWS re:Invent 2020 | Automate anything with AWS Systems Manager
- AWS re:Inforce 2022 | Automating patch management and compliance using AWS (NIS306)
- Supports You | Diving Deep into AWS Systems Manager

OPS02-BP04 Mechanisms exist to manage responsibilities and ownership

Understand the responsibilities of your role and how you contribute to business outcomes, as this understanding informs the prioritization of your tasks and why your role is important. This helps team members recognize needs and respond appropriately. When team members know their role, they can establish ownership, identify improvement opportunities, and understand how to influence or make appropriate changes.

Occasionally, a responsibility might not have a clear owner. In these situations, design a mechanism to resolve this gap. Create a well-defined escalation path to someone with the authority to assign ownership or plan to address the need.

Desired outcome: Teams within your organization have clearly-defined responsibilities that include how they are related to resources, actions to be performed, processes, and procedures. These responsibilities align to the team's responsibilities and goals, as well as the responsibilities of other teams. You document the routes of escalation in a consistent and discoverable manner and feed these decisions into documentation artifacts, such as responsibility matrices, team definitions, or wiki pages.

Common anti-patterns:

- The responsibilities of the team are ambiguous or poorly-defined.
- The team does not align roles with responsibilities.
- The team does not align its goals and objectives its responsibilities, which makes it difficult to measure success.
- Team member responsibilities do not align with the team and the wider organization.

- Your team does not keep responsibilities up-to-date, which makes them inconsistent with the tasks performed by the team.
- Escalation paths for determining responsibilities aren't defined or are unclear.
- Escalation paths have no single thread owner to ensure timely reponse.
- Roles, responsibilities, and escalation paths are not discoverable, and they are not readily available when required (for example, in response to an incident).

Benefits of establishing this best practice:

- When you understand who has responsibility or ownership, you can contact the proper team or team member to make a request or transition a task.
- To reduce the risk of inaction and unaddressed needs, you have identified a person who has the authority to assign responsibility or ownership.
- When you clearly define the scope of a responsibility, your team members gain autonomy and ownership.
- Your responsibilities inform the decisions you make, the actions you take, and your handoff activities to their proper owners.
- It's easy to identify abandoned responsibilities because you have a clear understanding of what falls outside of your team's responsibility, which helps you escalate for clarification.
- Teams avoid confusion and tension, and they can more adequately manage their workloads and resources.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify team members roles and responsibilities, and verify that they understand the expectations of their role. Make this information discoverable so that members of your organization can identify who they need to contact for specific needs, whether it's a team or individual. As organizations seek to capitalize on the opportunities to migrate and modernize on AWS, roles and responsibilities might also change. Keep your teams and their members aware of their responsibilities, and train them appropriately to carry out their tasks during this change.

Determine the role or team that should receive escalations to identify responsibility and ownership. This team can engage with various stakeholders to come to a decision. However, they should own the management of the decision making process.

Provide accessible mechanisms for members of your organization to discover and identify ownership and responsibility. These mechanisms teach them who to contact for specific needs.

Customer example

AnyCompany Retail recently completed a migration of workloads from an on-premises environment to their landing zone in AWS with a lift and shift approach. They performed an operations review to reflect on how they accomplish common operational tasks and verified that their existing responsibility matrix reflects operations in the new environment. When they migrated from on-premises to AWS, they reduced the infrastructure teams responsibilities relating to the hardware and physical infrastructure. This move also revealed new opportunities to evolve the operating model for their workloads.

While they identified, addressed, and documented the majority of responsibilities, they also defined escalation routes for any responsibilities that were missed or that may need to change as operations practices evolve. To explore new opportunities to standardize and improve efficiency across your workloads, provide access to operations tools like AWS Systems Manager and security tools like AWS Security Hub CSPM and Amazon GuardDuty. AnyCompany Retail puts together a review of responsibilities and strategy based on improvements they wants to address first. As the company adopts new ways of working and technology patterns, they update their responsibility matrix to match.

Implementation steps

1. Start with existing documentation. Some typical source documents might include:
    a. Responsibility or responsible, accountable, consulted, and informed (RACI) matrices
    b. Team definitions or wiki pages
    c. Service definitions and offerings
    d. Role or job descriptions
2. Review and host discussions on the documented responsibilities:
    a. Review with teams to identify misalignments between documented responsibilities and responsibilities the team typically performs.
    b. Discuss potential services offered by internal customers to identify gaps in expectations between teams.
3. Analysis and address the discrepancies.
4. Identify opportunities for improvement.

   a. Identify frequently-requested, resource-intensive requests, which are typically strong candidates for improvement.
   b. Look for best practices, understand patterns, follow prescriptive guidance, and simplify and standardize improvements.
   c. Record improvement opportunities, and track them to completion.
5. If a team doesn't already hold responsibility for managing and tracking the assignment of responsibilities, identify someone on the team to hold this responsibility.
6. Define a process for teams to request clarification of responsibility.
   a. Review the process, and verify that it is clear and simple to use.
   b. Make sure that someone owns and tracks escalations to their conclusion.
   c. Establish operational metrics to measure effectiveness.
   d. Create a feedback mechanisms to verify that teams can highlight improvement opportunities.
   e. Implement a mechanism for periodic review.
7. Document in a discoverable and accessible location.
   a. Wikis or documentation portal are common choices.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS01-BP06 Evaluate tradeoffs
- OPS03-BP02 Team members are empowered to take action when outcomes are at risk
- OPS03-BP03 Escalation is encouraged
- OPS03-BP07 Resource teams appropriately
- OPS09-BP01 Measure operations goals and KPIs with metrics
- OPS09-BP03 Review operations metrics and prioritize improvement
- OPS11-BP01 Have a process for continuous improvement

Related documents:

- AWS Whitepaper - Introduction to DevOps on AWS
- AWS Whitepaper - AWS Cloud Adoption Framework: Operations Perspective

- AWS Well-Architected Framework Operational Excellence - Workload level Operating model topologies
- AWS Prescriptive Guidance - Building your Cloud Operating Model
- AWS Prescriptive Guidance - Create a RACI or RASCI matrix for a cloud operating model
- AWS Cloud Operations & Migrations Blog - Delivering Business Value with Cloud Platform Teams
- AWS Cloud Operations & Migrations Blog - Why a Cloud Operating Model?
- AWS DevOps Blog - How organizations are modernizing for cloud operations

Related videos:

- AWS Summit Online - Cloud Operating Models for Accelerated Transformation
- AWS re:Invent 2023 - Future-proofing cloud security: A new operating model

OPS02-BP05 Mechanisms exist to request additions, changes, and exceptions

You can make requests to owners of processes, procedures, and resources. Requests include additions, changes, and exceptions. These requests go through a change management process. Make informed decisions to approve requests where viable and determined to be appropriate after an evaluation of benefits and risks.

Desired outcome:

- You can make requests to change processes, procedures, and resources based on assigned ownership.
- Changes are made in a deliberate manner, weighing benefits and risks.

Common anti-patterns:

- You must update the way you deploy your application, but there is no way to request a change to the deployment process from the operations team.
- The disaster recovery plan must be updated, but there is no identified owner to request changes to.

Benefits of establishing this best practice:

- Processes, procedures, and resources can evolve as requirements change.

- Owners can make informed decisions when to make changes.
- Changes are made in a deliberate manner.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To implement this best practice, you need to be able to request changes to processes, procedures, and resources. The change management process can be lightweight. Document the change management process.

Customer example

AnyCompany Retail uses a responsibility assignment (RACI) matrix to identify who owns changes for processes, procedures, and resources. They have a documented change management process that's lightweight and easy to follow. Using the RACI matrix and the process, anyone can submit change requests.

Implementation steps

1. Identify the processes, procedures, and resources for your workload and the owners for each. Document them in your knowledge management system.
   a. If you have not implemented OPS02-BP01 Resources have identified owners, OPS02-BP02 Processes and procedures have identified owners, or OPS02-BP03 Operations activities have identified owners responsible for their performance, start with those first.
2. Work with stakeholders in your organization to develop a change management process. The process should cover additions, changes, and exceptions for resources, processes, and procedures.
   a. You can use AWS Systems Manager Change Manager as a change management platform for workload resources.
3. Document the change management process in your knowledge management system.

Level of effort for the implementation plan: Medium. Developing a change management process requires alignment with multiple stakeholders across your organization.

Resources

Related best practices:

- OPS02-BP01 Resources have identified owners - Resources need identified owners before you build a change management process.
- OPS02-BP02 Processes and procedures have identified owners - Processes need identified owners before you build a change management process.
- OPS02-BP03 Operations activities have identified owners responsible for their performance - Operations activities need identified owners before you build a change management process.

Related documents:

- AWS Prescriptive Guidance - Foundation palybook for AWS large migrations: Creating RACI matrices
- Change Management in the Cloud Whitepaper

Related services:

- AWS Systems Manager Change Manager

OPS02-BP06 Responsibilities between teams are predefined or negotiated

Have defined or negotiated agreements between teams describing how they work with and support each other (for example, response times, service level objectives, or service-level agreements). Inter-team communications channels are documented. Understanding the impact of the teams' work on business outcomes and the outcomes of other teams and organizations informs the prioritization of their tasks and helps them respond appropriately.

When responsibility and ownership are undefined or unknown, you are at risk of both not addressing necessary activities in a timely fashion and of redundant and potentially conflicting efforts emerging to address those needs.

Desired outcome:

- Inter-team working or support agreements are agreed to and documented.
- Teams that support or work with each other have defined communication channels and response expectations.

Common anti-patterns:

- An issue occurs in production and two separate teams start troubleshooting independent of each other. Their siloed efforts extend the outage.
- The operations team needs assistance from the development team but there is no agreed to response time. The request is stuck in the backlog.

Benefits of establishing this best practice:

- Teams know how to interact and support each other.
- Expectations for responsiveness are known.
- Communications channels are clearly defined.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Implementing this best practice means that there is no ambiguity about how teams work with each other. Formal agreements codify how teams work together or support each other. Inter-team communication channels are documented.

Customer example

AnyCompany Retail's SRE team has a service level agreement with their development team. Whenever the development team makes a request in their ticketing system, they can expect a response within fifteen minutes. If there is a site outage, the SRE team takes lead in the investigation with support from the development team.

Implementation steps

1. Working with stakeholders across your organization, develop agreements between teams based on processes and procedures.
   a. If a process or procedure is shared between two teams, develop a runbook on how the teams will work together.
   b. If there are dependencies between teams, agree to a response SLA for requests.
2. Document responsibilities in your knowledge management system.

Level of effort for the implementation plan: Medium. If there are no existing agreements between teams, it can take effort to come to agreement with stakeholders across your organization.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners - Process ownership must be identified before setting agreements between teams.
- OPS02-BP03 Operations activities have identified owners responsible for their performance - Operations activities ownership must be identified before setting agreements between teams.

Related documents:

- AWS Executive Insights - Empowering Innovation with the Two-Pizza Team
- Introduction to DevOps on AWS - Two-Pizza Teams

OPS 3. How does your organizational culture support your business outcomes?

Provide support for your team members so that they can be more effective in taking action and supporting your business outcome.

Best practices

- OPS03-BP01 Provide executive sponsorship
- OPS03-BP02 Team members are empowered to take action when outcomes are at risk
- OPS03-BP03 Escalation is encouraged
- OPS03-BP04 Communications are timely, clear, and actionable
- OPS03-BP05 Experimentation is encouraged
- OPS03-BP06 Team members are encouraged to maintain and grow their skill sets
- OPS03-BP07 Resource teams appropriately

OPS03-BP01 Provide executive sponsorship

At the highest level, senior leadership acts as the executive sponsor to clearly set expectations and direction for the organization's outcomes, including evaluating its success. The sponsor advocates and drives adoption of best practices and evolution of the organization.

Desired outcome: Organizations that endeavor to adopt, transform, and optimize their cloud operations establish clear lines of leadership and accountability for desired outcomes. The organization understands each capability required by the organization to accomplish a new outcome and assigns ownership to functional teams for development. Leadership actively sets this direction, assigns ownership, takes accountability, and defines the work. As a result, individuals across the organization can mobilize, feel inspired, and actively work towards the desired objectives.

Common anti-patterns:

- There is a mandate for workload owners to migrate workloads to AWS without a clear sponsor and plan for cloud operations. This results in teams not consciously collaborating to improve and mature their operational capabilities. Lack of operational best practice standards overwhelm teams (such as operator-toil, on-calls, and technical debt), which constrains innovation.
- A new organization-wide goal has been set to adopt an emerging technology without providing leadership sponsor and strategy. Teams interpret goals differently, which causes confusion on where to focus efforts, why they matter, and how to measure impact. Consequently, the organization loses momentum in adopting the technology.

Benefits of establishing this best practice: When executive sponsorship clearly communicates and shares vision, direction, and goals, team members know what is expected of them. Individuals and teams begin to intensely focus effort in the same direction to accomplish defined objectives when leaders are actively engaged. As a result, the organization maximizes the ability to succeed. When you evaluate success, you can better identify barriers to success so that they can be addressed through intervention by the executive sponsor.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- At every phase of the cloud journey (migration, adoption, or optimization), success requires active involvement at the highest level of leadership with a designated executive sponsor. The

executive sponsor aligns the team's mindset, skillsets, and ways of working to the defined strategy.

- **Explain the *why*:** Bring clarity and explain the reasoning behind the vision and strategy.
- **Set expectations:** Define and publish goals for your organizations, including how progress and success are measured.
- **Track achievement of goals:** Measure the incremental achievement of goals regularly (not just completion of tasks). Share the results so that appropriate action can be taken if outcomes are at risk.
- **Provide the resources necessary to achieve your goals:** Bring people and teams together to collaborate and build the right solutions that bring about the defined outcomes. This reduces or eliminates organizational friction.
- **Advocate for your teams:** Remain engaged with your teams so that you understand their performance and whether there are external factors affecting them. Identify obstacles that are impeding your teams progress. Act on behalf of your teams to help address obstacles and remove unnecessary burdens. When your teams are impacted by external factors, reevaluate goals and adjust targets as appropriate.
- **Drive adoption of best practices:** Acknowledge best practices that provide quantifiable benefits, and recognize the creators and adopters. Encourage further adoption to magnify the benefits achieved.
- **Encourage evolution of your teams:** Create a culture of continual improvement, and proactively learn from progress made as well as failures. Encourage both personal and organizational growth and development. Use data and anecdotes to evolve the vision and strategy.

Customer example

AnyCompany Retail is in the process of business transformation through rapid reinvention of customer experiences, enhancement of productivity, and acceleration of growth through generative AI.

Implementation steps

1. Establish single-threaded leadership, and assign a primary executive sponsor to lead and drive the transformation.

2. Define clear business outcomes of your transformation, and assign ownership and accountability. Empower the primary executive with the authority to lead and make critical decisions.
3. Verify that your transformational strategy is very clear and communicated widely by the executive sponsor to every level of the organization.
    a. Establish clearly defined business objectives for IT and cloud initiatives.
    b. Document key business metrics to drive IT and cloud transformation.
    c. Communicate the vision consistently to all teams and individuals responsible for parts of the strategy.
4. Develop communication planning matrices that specify what message needs to be delivered to specified leaders, managers, and individual contributors. Specify the person or team that should deliver this message.
    a. Fulfill communications plans consistently and reliably.
    b. Set and manage expectations through in-person events on a regular basis.
    c. Accept feedback on the effectiveness of communications, and adjust the communications and plan accordingly.
    d. Schedule communication events to proactively understand challenges from teams, and establish a consistent feedback loop that allows for correcting course where necessary.
5. Actively engage each initiative from a leadership perspective to verify that all impacted teams understand the outcomes they are accountable to achieve.
6. At every status meeting, executive sponsors should look for blockers, inspect established metrics, anecdotes, or feedback from the teams, and measure progress towards objectives.

Level of effort for the implementation plan Medium

Resources

Related best practices:

- OPS03-BP04 Communications are timely, clear, and actionable
- OP11-BP01 Have a process for continuous improvement
- OPS11-BP07 Perform operations metrics reviews

Related documents:

- Untangling Your Organisational Hairball: Highly Aligned
- The Living Transformation: Pragmatically approaching changes
- Becoming a Future-Ready Enterprise
- 7 Pitfalls to Avoid When Building a CCOE
- Navigating the Cloud: Key Performance Indicators for Success

Related videos:

- AWS re:Invent 2023: A leader's guide to generative AI: Using history to shape the future (SEG204)

Related examples:

- Prosci: Primary Sponsor's Role & Importance

OPS03-BP02 Team members are empowered to take action when outcomes are at risk

A cultural behavior of ownership instilled by leadership results in any employee feeling empowered to act on behalf of the entire company beyond their defined scope of role and accountability. Employees can act to proactively identify risks as they emerge and take appropriate action. Such a culture allows employees to make high value decisions with situational awareness.

For example, Amazon uses Leadership Principles as the guidelines to drive desired behavior for employees to move forward in situations, solve problems, deal with conflict, and take action.

Desired outcome: Leadership has influenced a new culture that allows individuals and teams to make critical decisions, even at lower levels of the organization (as long as decisions are defined with auditable permissions and safety mechanisms). Failure is not discouraged, and teams iteratively learn to improve their decision-making and responses to tackle similar situations going forward. If someone's actions result in an improvement that can benefit other teams, they proactively share knowledge from such actions. Leadership measures operational improvements and incentivizes the individual and organization for adoption of such patterns.

Common anti-patterns:

- There isn't clear guidance or mechanisms in an organization for what to do when a risk is identified. For example, when an employee notices a phishing attack, they fail to report to the security team, resulting in a large portion of the organization falling for the attack. This causes a data breach.

- Your customers complain about service unavailability, which primarily stems from failed deployments. Your SRE team is responsible for the deployment tool, and an automated rollback for deployments is in their long-term roadmap. In a recent application rollout, one of the engineers devised a solution to automate rolling back their application to a previous version. Though their solution can become the pattern for SRE teams, other teams do not adopt, as there is no process to track such improvements. The organization continues to be plagued with failed deployments impacting customers and causing further negative sentiment.
- In order to stay compliant, your infosec team oversees a long-established process to rotate shared SSH keys regularly on behalf of operators connecting to their Amazon EC2 Linux instances. It takes several days for the infosec teams to complete rotating keys, and you are blocked from connecting to those instances. No one inside or outside of infosec suggests using other options on AWS to achieve the same result.

Benefits of establishing this best practice: By decentralizing authority to make decisions and empowering your teams to decide key decisions, you are able to address issues more quickly with increasing success rates. In addition, teams start to realize a sense of ownership, and failures are acceptable. Experimentation becomes a cultural mainstay. Managers and directors do not feel as though they are micro-managed through every aspect of their work.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

1. Develop a culture where it is expected that failures can occur.
2. Define clear ownership and accountability for various functional areas within the organization.
3. Communicate ownership and accountability to everyone so that individuals know who can help them facilitate decentralized decisions.
4. Define your one-way and two-way door decisions to help individuals know when they do need to escalate to higher levels of leadership.
5. Create organizational awareness that all employees are empowered to take action at various levels when outcomes are at risk. Provide your team members documentation of governance, permission-levels, tools, and opportunities to practice the skills necessary to respond effectively.
6. Give your team members the opportunity to practice the skills necessary to respond to various decisions. Once decision levels are defined, perform game days to verify that all individual contributors understand and can demonstrate the process.

   a. Provide alternative safe environments where processes and procedures can be tested and trained upon.
   b. Acknowledge and create awareness that team members have authority to take action when the outcome has a predefined level of risk.
   c. Define the authority of your team members to take action by assigning permissions and access to the workloads and components they support.
7. Provide ability for teams to share their learnings (operational successes and failures).
8. Empower teams to challenge the status quo, and provide mechanisms to track and measure improvements, as well as their impact to the organization.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS01-BP06 Evaluate tradeoffs while managing benefits and risks
- OPS02-BP05 Mechanisms exist to identify responsibility and ownership

Related documents:

- AWS Blog Post | The agile enterprise
- AWS Blog Post | Measuring success : A paradox and a plan
- AWS Blog Post | Letting go : Enabling autonomy in teams
- Centralize or Decentralize?

Related videos:

- re:Invent 2023 | How to not sabotage your transformation (SEG201)
- re:Invent 2021 | Amazon Builders' Library: Operational Excellence at Amazon
- Centralization vs. Decentralization

Related examples:

- Using architectural decision records to streamline technical decision-making for a software development project

OPS03-BP03 Escalation is encouraged

Team members are encouraged by leadership to escalate issues and concerns to higher-level decision makers and stakeholders if they believe desired outcomes are at risk and expected standards are not met. This is a feature of the organization's culture and is driven at all levels. Escalation should be done early and often so that risks can be identified and prevented from causing incidents. Leadership does not reprimand individuals for escalating an issue.

Desired outcome: Individuals throughout the organization are comfortable to escalate problems to their immediate and higher levels of leadership. Leadership has deliberately and consciously established expectations that their teams should feel safe to escalate any issue. A mechanism exists to escalate issues at each level within the organization. When employees escalate to their manager, they jointly decide the level of impact and whether the issue should be escalated. In order to initiate an escalation, employees are required to include a recommended work plan to address the issue. If direct management does not take timely action, employees are encouraged to take issues to the highest level of leadership if they feel strongly that the risks to the organization warrant the escalation.

Common anti-patterns:

- Executive leaders do not ask enough probing questions during your cloud transformation program status meeting to find where issues and blockers are occurring. Only good news is presented as status. The CIO has made it clear that she only likes to hear good news, as any challenges brought up make the CEO think that the program is failing.
- You are a cloud operations engineer and you notice that the new knowledge management system is not being widely adopted by application teams. The company invested one year and several million dollars to implement this new knowledge management system, but people are still authoring their runbooks locally and sharing them on an organizational cloud share, making it difficult to find knowledge pertinent to supported workloads. You try to bring this to leadership's attention, because consistent use of this system can enhance operational efficiency. When you bring this to the director who lead the implementation of the knowledge management system, she reprimands you because it calls the investment into question.
- The infosec team responsible for hardening compute resources has decided to put a process in place that requires performing the scans necessary to ensure that EC2 instances are fully

secured before the compute team releases the resource for use. This has created a time delay of an additional week for resources to be deployed, which breaks their SLA. The compute team is afraid to escalate this to the VP over cloud because this makes the VP of information security look bad.

Benefits of establishing this best practice:

Complex or critical issues are addressed before they impact the business. Less time is wasted. Risks are minimized. Teams become more proactive and results focused when solving problems.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The willingness and ability to escalate freely at every level in the organization is an organizational and cultural foundation that should be consciously developed through emphasized training, leadership communications, expectation setting, and the deployment of mechanisms throughout the organization at every level.

Implementation steps

1. Define policies, standards, and expectations for your organization.
   a. Ensure wide adoption and understanding of policies, expectations, and standards.
2. Encourage, train, and empower workers for early and frequent escalation when standards are not met.
3. Organizationally acknowledge that early and frequent escalation is the best practice. Accept that escalations may prove to be unfounded, and that it is better to have the opportunity to prevent an incident then to miss that opportunity by not escalating.
   a. Build a mechanism for escalation (like an Andon cord system).
   b. Have documented procedures defining when and how escalation should occur.
   c. Define the series of people with increasing authority to take or approve action, as well as each stakeholder's contact information.
4. When escalation occurs, it should continue until the team member is satisfied that the risk has been mitigated through actions driven from leadership.
   a. Escalations should include:
      i. Description of the situation, and the nature of the risk

ii. Criticality of the situation

iii. Who or what is impacted

iv. How great the impact is

v. Urgency if impact occurs

vi. Suggested remedies and plans to mitigate

b. Protect employees who escalate. Have policy that protects team members from retribution if they escalate around a non-responsive decision maker or stakeholder. Have mechanisms in place to identify if this is occurring and respond appropriately.

5. Encourage a culture of continuous improvement feedback loops in everything that the organization produces. Feedback loops act as minor escalations to individuals responsible, and they identify improvement opportunities, even when escalation is not needed. Continuous improvement cultures force everyone to be more proactive.
6. Leadership should periodically reemphasize the policies, standards, mechanisms, and the desire for open escalation and continuous feedback loops without retribution.

Level of effort for the Implementation Plan: Medium

Resources

Related best practices:

- OPS02-BP05 Mechanisms exist to request additions, changes, and exceptions

Related documents:

- How do you foster a culture of continuous improvement and learning from Andon and escalation systems?
- The Andon Cord (IT Revolution)
- AWS DevOps Guidance | Establish clear escalation paths and encourage constructive disagreement

Related videos:

- Jeff Bezos on how to make decisions (& increase velocity)
- Toyota Product System: Stopping Production, a Button, and an Andon Electric Board

- Andon Cord in LEAN Manufacturing

Related examples:

- Working with escalation plans in Incident Manager

OPS03-BP04 Communications are timely, clear, and actionable

Leadership is responsible for the creation of strong and effective communications, especially when the organization adopts new strategies, technologies, or ways of working. Leaders should set expectations for all staff to work towards the company objectives. Devise communication mechanisms that create and maintain awareness among the teams responsible for running plans that are funded and sponsored by leadership. Make use of cross-organizational diversity, and listen attentively to multiple unique perspectives. Use this perspective to increase innovation, challenge your assumptions, and reduce the risk of confirmation bias. Foster inclusion, diversity, and accessibility within your teams to gain beneficial perspectives.

Desired outcome: Your organization designs communication strategies to address the impact of change to the organization. Teams remain informed and motivated to continue working with one another rather than against each other. Individuals understand how important their role is to achieve the stated objectives. Email is only a passive mechanism for communications and used accordingly. Management spends time with their individual contributors to help them understand their responsibility, the tasks to complete, and how their work contributes to the overall mission. When necessary, leaders engage people directly in smaller venues to convey messages and verify that these messages are being delivered effectively. As a result of good communications strategies, the organization performs at or above the expectations of leadership. Leadership encourages and seeks diverse opinions within and across teams.

Common anti-patterns:

- Your organization has a five year plan to migrate all workloads to AWS. The business case for cloud includes the modernization of 25% of all workloads to take advantage of serverless technology. The CIO communicates this strategy to direct reports and expects each leader to cascade this presentation to managers, directors, and individual contributors without any in-person communication. The CIO steps back and expects his organization to perform the new strategy.
- Leadership does not provide or use a mechanism for feedback, and an expectation gap grows, which leads to stalled projects.

- You are asked to make a change to your security groups, but you are not given any details of what change needs to be made, what the impact of the change could be on all the workloads, and when it should happen. The manager forwards an email from the VP of InfoSec and adds the message "Make this happen."
- Changes were made to your migration strategy that reduce the planned modernization number from 25% to 10%. This has downstream effects on the operations organization. They were not informed of this strategic change and thus, they are not ready with enough skilled capacity to support a greater number of workloads lifted and shifted into AWS.

Benefits of establishing this best practice:

- Your organization is well-informed on new or changed strategies, and they act accordingly with strong motivation to help each other achieve the overall objectives and metrics set by leadership.
- Mechanisms exist and are used to provide timely notice to team members of known risks and planned events.
- New ways of working (including changes to people or the organization, processes, or technology), along with required skills, are more effectively adopted by the organization, and your organization realizes business benefits more quickly.
- Team members have the necessary context of the communications being received, and they can be more effective in their jobs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To implement this best practice, you must work with stakeholders across your organization to agree to communication standards. Publicize those standards to your organization. For any significant IT transitions, an established planning team can more successfully manage the impact of change to its people than an organization that ignores this practice. Larger organizations can be more challenging when managing change because it's critical to establish strong buy-in on a new strategy with all individual contributors. In the absence of such a transition planning team, leadership holds 100% of the responsibility for effective communications. When establishing a transition planning team, assign team members to work with all organizational leadership to define and manage effective communications at every level.

Customer example

AnyCompany Retail signed up for AWS Enterprise Support and depends on other third-party providers for its cloud operations. The company uses chat and chatops as their main communication medium for operational activities. Alerts and other information populate specific channels. When someone must act, they clearly state the desired outcome, and in many cases, they receive a runbook or playbook to use. They schedule major changes to production systems with a change calendar.

Implementation steps

1. Establish a core team within the organization that has accountability to build and initiate communication plans for changes that happen at multiple levels within the organization.
2. Institute single-threaded ownership to achieve oversight. Give individual teams the ability to innovate independently, and balance the use of consistent mechanisms, which allows for the right level of inspection and directional vision.
3. Work with stakeholders across your organization to agree to communication standards, practices, and plans.
4. Verify that the core communications team collaborates with organizational and program leadership to craft messages to appropriate staff on behalf of leaders.
5. Build strategic communication mechanisms to manage change through announcements, shared calendars, all-hands meetings, and in-person or one-on-one methods so that team members have proper expectations on the actions they should take.
6. Provide necessary context, details, and time (when possible) to determine if action is necessary. When action is needed, provide the required action and its impact.
7. Implement tools that facilitate tactical communications, like internal chat, email, and knowledge management.
8. Implement mechanisms to measure and verify that all communications lead to desired outcomes.
9. Establish a feedback loop that measures the effectiveness of all communications, especially when communications are related to resistance to changes throughout the organization.
10. For all AWS accounts, establish alternate contacts for billing, security, and operations. Ideally, each contact should be an email distribution as opposed to a specific individual contact.
11. Establish an escalation and reverse escalation communication plan to engage with your internal and external teams, including AWS support and other third-party providers.
12. Initiate and perform communication strategies consistently throughout the life of each transformation program.

13. Prioritize actions that are repeatable where possible to safely automate at scale.
14. When communications are required in scenarios with automated actions, the communication's purpose should be to inform teams, for auditing, or a part of the change management process.
15. Analyze communications from your alert systems for false positives or alerts that are constantly created. Remove or change these alerts so that they start when human intervention is required. If an alert is initiated, provide a runbook or playbook.
    a. You can use AWS Systems Manager Documents to build playbooks and runbooks for alerts.
16. Mechanisms are in place to provide notification of risks or planned events in a clear and actionable way with enough notice to allow appropriate responses. Use email lists or chat channels to send notifications ahead of planned events.
    a. AWS Chatbot can be used to send alerts and respond to events within your organizations messaging platform.
17. Provide an accessible source of information where planned events can be discovered. Provide notifications of planned events from the same system.
    a. AWS Systems Manager Change Calendar can be used to create change windows when changes can occur. This provides team members notice when they can make changes safely.
18. Monitor vulnerability notifications and patch information to understand vulnerabilities in the wild and potential risks associated to your workload components. Provide notification to team members so that they can act.
    a. You can subscribe to AWS Security Bulletins to receive notifications of vulnerabilities on AWS.
19. **Seek diverse opinions and perspectives:** Encourage contributions from everyone. Give communication opportunities to under-represented groups. Rotate roles and responsibilities in meetings.
    a. **Expand roles and responsibilities:** Provide opportunities for team members to take on roles that they might not otherwise. They can gain experience and perspective from the role and from interactions with new team members with whom they might not otherwise interact. They can also bring their experience and perspective to the new role and team members they interact with. As perspective increases, identify emergent business opportunities or new opportunities for improvement. Rotate common tasks between members within a team that others typically perform to understand the demands and impact of performing them.
    b. **Provide a safe and welcoming environment:** Establish policy and controls that protect the mental and physical safety of team members within your organization. Team members should be able to interact without fear of reprisal. When team members feel safe and welcome, they are more likely to be engaged and productive. The more diverse your organization, the better

your understanding can be of the people you support, including your customers. When your team members are comfortable, feel free to speak, and are confident they are heard, they are more likely to share valuable insights (for example, marketing opportunities, accessibility needs, unserved market segments, and unacknowledged risks in your environment).

c. **Encourage team members to participate fully:** Provide the resources necessary for your employees to participate fully in all work related activities. Team members that face daily challenges develop skills for working around them. These uniquely-developed skills can provide significant benefit to your organization. Support team members with necessary accommodations to increase the benefits you can receive from their contributions.

Resources

Related best practices:

- OPS03-BP01 Provide executive sponsorship
- OPS07-BP03 Use runbooks to perform procedures
- OPS07-BP04 Use playbooks to investigate issues

Related documents:

- AWS Blog post | Accountability and empowerment are key to high-performing agile organizations
- AWS Executive Insights | Learn to scale innovation, not complexity | Single-threaded Leaders
- AWS Security Bulletins
- Open CVE
- Support App in Slack to Manage Support Cases
- Manage AWS resources in your Slack channels with Amazon Q Developer in chat applications

Related services:

- Amazon Q Developer in chat applications
- AWS Systems Manager Change Calendar
- AWS Systems Manager Documents

OPS03-BP05 Experimentation is encouraged

Experimentation is a catalyst for turning new ideas into products and features. It accelerates learning and keeps team members interested and engaged. Team members are encouraged to experiment often to drive innovation. Even when an undesired result occurs, there is value in knowing what not to do. Team members are not punished for successful experiments with undesired results.

Desired outcome:

- Your organization encourages experimentation to foster innovation.
- Experiments are used as an opportunity to learn.

Common anti-patterns:

- You want to run an A/B test but there is no mechanism to run the experiment. You deploy a UI change without the ability to test it. It results in a negative customer experience.
- Your company only has a stage and production environment. There is no sandbox environment to experiment with new features or products so you must experiment within the production environment.

Benefits of establishing this best practice:

- Experimentation drives innovation.
- You can react faster to feedback from users through experimentation.
- Your organization develops a culture of learning.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Experiments should be run in a safe manner. Leverage multiple environments to experiment without jeopardizing production resources. Use A/B testing and feature flags to test experiments. Provide team members the ability to conduct experiments in a sandbox environment.

Customer example

AnyCompany Retail encourages experimentation. Team members can use 20% of their work week to experiment or learn new technologies. They have a sandbox environment where they can innovate. A/B testing is used for new features to validate them with real user feedback.

Implementation steps

1. Work with leadership across your organization to support experimentation. Team members should be encouraged to conduct experiments in a safe manner.
2. Provide your team members with an environment where they can safely experiment. They must have access to an environment that is like production.
   a. You can use a separate AWS account to create a sandbox environment for experimentation. AWS Control Tower can be used to provision these accounts.
3. Use feature flags and A/B testing to experiment safely and gather user feedback.
   a. AWS AppConfig Feature Flags provides the ability to create feature flags.
   b. You can use AWS Lambda versions to deploy a new version of a function for beta testing.

Level of effort for the implementation plan: High. Providing team members with an environment to experiment in and a safe way to conduct experiments can require significant investment. You may also need to modify application code to use feature flags or support A/B testing.

Resources

Related best practices:

- OPS11-BP02 Perform post-incident analysis - Learning from incidents is an important driver for innovation along with experimentation.
- OPS11-BP03 Implement feedback loops - Feedback loops are an important part of experimentation.

Related documents:

- An Inside Look at the Amazon Culture: Experimentation, Failure, and Customer Obsession
- Best practices for creating and managing sandbox accounts in AWS
- Create a Culture of Experimentation Enabled by the Cloud
- Enabling experimentation and innovation in the cloud at SulAmérica Seguros
- Experiment More, Fail Less

- Organizing Your AWS Environment Using Multiple Accounts - Sandbox OU
- Using AWS AppConfig Feature Flags

Related videos:

- AWS On Air ft. Amazon CloudWatch Evidently | AWS Events
- AWS On Air San Fran Summit 2022 ft. AWS AppConfig Feature Flags integration with Jira
- AWS re:Invent 2022 - A deployment is not a release: Control your launches w/feature flags (BOA305-R)
- Programmatically Create an AWS account with AWS Control Tower
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations

Related examples:

- AWS Innovation Sandbox
- End-to-end Personalization 101 for E-Commerce

Related services:

- Amazon CloudWatch Evidently
- AWS AppConfig
- AWS Control Tower

OPS03-BP06 Team members are encouraged to maintain and grow their skill sets

Teams must grow their skill sets to adopt new technologies, and to support changes in demand and responsibilities in support of your workloads. Growth of skills in new technologies is frequently a source of team member satisfaction and supports innovation. Support your team members' pursuit and maintenance of industry certifications that validate and acknowledge their growing skills. Cross train to promote knowledge transfer and reduce the risk of significant impact when you lose skilled and experienced team members with institutional knowledge. Provide dedicated structured time for learning.

AWS provides resources, including the AWS Getting Started Resource Center, AWS Blogs, AWS Online Tech Talks, AWS Events and Webinars, and the AWS Well-Architected Labs, that provide guidance, examples, and detailed walkthroughs to educate your teams.

Resources such as Support, (AWS re:Post, Support Center), and AWS Documentation help remove technical roadblocks and improve operations. Reach out to Support through Support Center for help with your questions.

AWS also shares best practices and patterns that we have learned through the operation of AWS in The Amazon Builders' Library and a wide variety of other useful educational material through the AWS Blog and The Official AWS Podcast.

AWS Training and Certification includes free training through self-paced digital courses, along with learning plans by role or domain. You can also register for instructor-led training to further support the development of your teams' AWS skills.

Desired outcome: Your organization constantly evaluates skill gaps and closes them with structured budget and investment. Teams encourage and incentivize their members with upskilling activities such as acquiring leading industry certifications. Teams take advantage of dedicated cross-sharing knowledge programs such as lunch-and-learns, immersion days, hackathons, and gamedays. Your organization's keeps its knowledge systems up-to-date and relevant to cross-train team members, including new-hire onboarding trainings.

Common anti-patterns:

- In the absence of a structured training program and budget, teams experience uncertainty as they try to keep pace with technology evolution, which results in increased attrition.
- As part of migrating to AWS, your organization demonstrates skill gaps and varying cloud fluency amongst teams. Without an effort to upskill, teams find themselves overtasked with legacy and inefficient management of the cloud environment, which causes increased operator toil. This burn out increases employee dissatisfaction.

Benefits of establishing this best practice: When your organization consciously invests in improving the skills of its teams, it also helps accelerate and scale cloud adoption and optimization. Targeted learning programs drive innovation and build operational ability for teams to be prepared to handle events. Teams consciously invest in the implementation and evolution of best practices. Team morale is high, and team members value their contribution to the business.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To adopt new technologies, fuel innovation, and keep pace with changes in demand and responsibilities to support your workloads, continually invest in the professional growth of your teams.

Implementation steps

1. **Use structured cloud advocacy programs:** AWS Skills Guild provides consultative training to increase cloud skill confidence and ignite a culture of continuous learning.
2. **Provide resources for education:** Provide dedicated, structured time and access to training materials and lab resources, and support participation in conferences and access to professional organizations that provide opportunities for learning from both educators and peers. Provide your junior team members with access to senior team members as mentors, or allow the junior team members to shadow their seniors' work and be exposed to their methods and skills. Encourage learning about content not directly related to work in order to have a broader perspective.
3. **Encourage use of expert technical resources:** Leverage resources such as AWS re:Post to get access to curated knowledge and vibrant community.
4. **Build and maintain an up-to-date knowledge repository:** Use knowledge sharing platforms such as wikis and runbooks. Create your own reusable expert knowledge source with AWS re:Post Private to streamline collaboration, improve productivity, and accelerate employee onboarding.
5. **Team education and cross-team engagement:** Plan for the continuing education needs of your team members. Provide opportunities for team members to join other teams (temporarily or permanently) to share skills and best practices benefiting your entire organization.
6. **Support pursuit and maintenance of industry certifications:** Support your team members in the acquisition and maintenance of industry certifications that validate what they have learned and acknowledge their accomplishments.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS03-BP01 Provide executive sponsorship

- OPS11-BP04 Perform knowledge management

Related documents:

- AWS Whitepaper | Cloud Adoption Framework: People Perspective
- Investing in continuous learning to grow your organization's future
- AWS Skills Guild
- AWS Training and Certification
- Support
- AWS re:Post
- AWS Getting Started Resource Center
- AWS Blogs
- AWS Cloud Compliance
- AWS Documentation
- The Official AWS Podcast.
- AWS Online Tech Talks
- AWS Events and Webinars
- AWS Well-Architected Labs
- The Amazon Builders' Library

Related videos:

- AWS re:Invent 2023 | Reskilling at the speed of cloud: Turning employees into entrepreneurs
- WS re:Invent 2023 | Building a culture of curiosity through gamification

OPS03-BP07 Resource teams appropriately

Provision the right amount of proficient team members, and provide tools and resources to support your workload needs. Overburdening team members increases the risk of human error. Investments in tools and resources, such as automation, can scale the effectiveness of your team and help them support a greater number of workloads without requiring additional capacity.

Desired outcome:

- You have appropriately staffed your team to gain the skillsets needed for them to operate workloads in AWS in accordance with your migration plan. As your team has scaled itself up during the course of your migration project, they have gained proficiency in the core AWS technologies that the business plans to use when migrating or modernizing their applications.
- You have carefully aligned your staffing plan to make efficient use of resources by leveraging automation and workflow. A smaller team can now manage more infrastructure on behalf of the application development teams.
- With shifting operational priorities, any resource staffing constraints are proactively identified to protect the success of business initiatives.
- Operational metrics that report operational toil (such as on-call fatigue or excessive paging) are reviewed to verify that staff are not overwhelmed.

Common anti-patterns:

- Your staff have not ramped up on AWS skills as you close in on your multi-year cloud migration plan, which risks support of the workloads and lowers employee morale.
- Your entire IT organization is shifting into agile ways of working. The business is prioritizing the product portfolio and setting metrics for what features need to be developed first. Your agile process does not require teams to assign story points to their work plans. As a result, it is impossible to know the level of capacity required for the next amount of work, or if you have the right skills assigned to the work.
- You are having an AWS partner migrate your workloads, and you don't have a support transition plan for your teams once the partner completes the migration project. Your teams struggle to efficiently and effectively support the workloads.

Benefits of establishing this best practice: You have appropriately-skilled team members available in your organization to support the workloads. Resource allocation can adapt to shifting priorities without impacting performance. The result is teams being proficient at supporting workloads while maximizing time to focus on innovating for customers, which in turn raises employee satisfaction.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Resource planning for your cloud migration should occur at an organizational level that aligns to your migration plan, as well as the desired operating model being implemented to support your

new cloud environment. This should include understanding which cloud technologies are deployed for the business and application development teams. Infrastructure and operations leadership should plan for skills gap analysis, training, and role definition for engineers who are leading cloud adoption.

Implementation steps

1. Define success criteria for team's success with relevant operational metrics such as staff productivity (for example, cost to support a workload or operator hours spent during incidents).
2. Define resource capacity planning and inspection mechanisms to verify that the right balance of qualified capacity is available when needed and can be adjusted over time.
3. Create mechanisms (for example, sending a monthly survey to teams) to understand work-related challenges that impact teams (like increasing responsibilities, changes in technology, loss of personnel, or increase in customers supported).
4. Use these mechanisms to engage with teams and spot trends that may contribute to employee productivity challenges. When your teams are impacted by external factors, reevaluate goals and adjust targets as appropriate. Identify obstacles that are impeding your team's progress.
5. Regularly review if your currently-provisioned resources are still sufficient, of if additional resources are needed, and make appropriate adjustments to support teams.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS03-BP06 Team members are encouraged to maintain and grow their skill sets
- OPS09-BP03 Review operations metrics and prioritize improvement
- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP07 Automate responses to events

Related documents:

- AWS Cloud Adoption Framework: People Perspective
- Becoming a Future-Ready Enterprise
- Prioritize your Employees' Skills to Drive Business Growth

- High performing organization - the Amazon Two-Pizza team
- How Cloud-Mature Enterprises Succeed

Prepare

Questions

- OPS 4. How do you implement observability in your workload?
- OPS 5. How do you reduce defects, ease remediation, and improve flow into production?
- OPS 6. How do you mitigate deployment risks?
- OPS 7. How do you know that you are ready to support a workload?

OPS 4. How do you implement observability in your workload?

Implement observability in your workload so that you can understand its state and make data-driven decisions based on business requirements.

Best practices

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement distributed tracing

OPS04-BP01 Identify key performance indicators

Implementing observability in your workload starts with understanding its state and making data-driven decisions based on business requirements. One of the most effective ways to ensure alignment between monitoring activities and business objectives is by defining and monitoring key performance indicators (KPIs).

Desired outcome: Efficient observability practices that are tightly aligned with business objectives, ensuring that monitoring efforts are always in service of tangible business outcomes.

Common anti-patterns:

- Undefined KPIs: Working without clear KPIs can lead to monitoring too much or too little, missing vital signals.
- Static KPIs: Not revisiting or refining KPIs as the workload or business objectives evolve.
- Misalignment: Focusing on technical metrics that don't correlate directly with business outcomes or are harder to correlate with real-world issues.

Benefits of establishing this best practice:

- Ease of issue identification: Business KPIs often surface issues more clearly than technical metrics. A dip in a business KPI can pinpoint a problem more effectively than sifting through numerous technical metrics.
- Business alignment: Ensures that monitoring activities directly support business objectives.
- Efficiency: Prioritize monitoring resources and attention on metrics that matter.
- Proactivity: Recognize and address issues before they have broader business implications.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To effectively define workload KPIs:

1. **Start with business outcomes:** Before diving into metrics, understand the desired business outcomes. Is it increased sales, higher user engagement, or faster response times?
2. **Correlate technical metrics with business objectives:** Not all technical metrics have a direct impact on business outcomes. Identify those that do, but it's often more straightforward to identify an issue using a business KPI.
3. **Use Amazon CloudWatch:** Employ CloudWatch to define and monitor metrics that represent your KPIs.
4. **Regularly review and update KPIs:** As your workload and business evolve, keep your KPIs relevant.
5. **Involve stakeholders:** Involve both technical and business teams in defining and reviewing KPIs.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- the section called “OPS04-BP02 Implement application telemetry”
- the section called “OPS04-BP03 Implement user experience telemetry”
- the section called “OPS04-BP04 Implement dependency telemetry”
- the section called “OPS04-BP05 Implement distributed tracing”

Related documents:

- AWS Observability Best Practices
- CloudWatch User Guide
- AWS Observability Skill Builder Course

Related videos:

- Developing an observability strategy

Related examples:

- One Observability Workshop

OPS04-BP02 Implement application telemetry

Application telemetry serves as the foundation for observability of your workload. It's crucial to emit telemetry that offers actionable insights into the state of your application and the achievement of both technical and business outcomes. From troubleshooting to measuring the impact of a new feature or ensuring alignment with business key performance indicators (KPIs), application telemetry informs the way you build, operate, and evolve your workload.

Metrics, logs, and traces form the three primary pillars of observability. These serve as diagnostic tools that describe the state of your application. Over time, they assist in creating baselines and identifying anomalies. However, to ensure alignment between monitoring activities and business objectives, it's pivotal to define and monitor KPIs. Business KPIs often make it easier to identify issues compared to technical metrics alone.

Other telemetry types, like real user monitoring (RUM) and synthetic transactions, complement these primary data sources. RUM offers insights into real-time user interactions, whereas synthetic transactions simulate potential user behaviors, helping detect bottlenecks before real users encounter them.

Desired outcome: Derive actionable insights into the performance of your workload. These insights allow you to make proactive decisions about performance optimization, achieve increased workload stability, streamline CI/CD processes, and utilize resources effectively.

Common anti-patterns:

- **Incomplete observability:** Neglecting to incorporate observability at every layer of the workload, resulting in blind spots that can obscure vital system performance and behavior insights.
- **Fragmented data view:** When data is scattered across multiple tools and systems, it becomes challenging to maintain a holistic view of your workload's health and performance.
- **User-reported issues:** A sign that proactive issue detection through telemetry and business KPI monitoring is lacking.

Benefits of establishing this best practice:

- **Informed decision-making:** With insights from telemetry and business KPIs, you can make data-driven decisions.
- **Improved operational efficiency:** Data-driven resource utilization leads to cost-effectiveness.
- **Enhanced workload stability:** Faster detection and resolution of issues leading to improved uptime.
- **Streamlined CI/CD processes:** Insights from telemetry data facilitate refinement of processes and reliable code delivery.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To implement application telemetry for your workload, use AWS services like Amazon CloudWatch and AWS X-Ray. Amazon CloudWatch provides a comprehensive suite of monitoring tools, allowing you to observe your resources and applications in AWS and on-premises environments. It collects, tracks, and analyzes metrics, consolidates and monitors log data, and responds to changes in your

resources, enhancing your understanding of how your workload operates. In tandem, AWS X-Ray lets you trace, analyze, and debug your applications, giving you a deep understanding of your workload's behavior. With features like service maps, latency distributions, and trace timelines, AWS X-Ray provides insights into your workload's performance and the bottlenecks affecting it.

Implementation steps

1. **Identify what data to collect:** Ascertain the essential metrics, logs, and traces that would offer substantial insights into your workload's health, performance, and behavior.
2. **Deploy the CloudWatch agent:** The CloudWatch agent is instrumental in procuring system and application metrics and logs from your workload and its underlying infrastructure. The CloudWatch agent can also be used to collect OpenTelemetry or X-Ray traces and send them to X-Ray.
3. **Implement anomaly detection for logs and metrics:** Use CloudWatch Logs anomaly detection and CloudWatch Metrics anomaly detection to automatically identify unusual activities in your application's operations. These tools use machine learning algorithms to detect and alert on anomalies, which enanhces your monitoring capabilities and speeds up response time to potential disruptions or security threats. Set up these features to proactively manage application health and security.
4. **Secure sensitive log data:** Use Amazon CloudWatch Logs data protection to mask sensitive information within your logs. This feature helps maintain privacy and compliance through automatic detection and masking of sensitive data before it is accessed. Implement data masking to securely handle and protect sensitive details such as personally identifiable information (PII).
5. **Define and monitor business KPIs:** Establish custom metrics that align with your business outcomes.
6. **Instrument your application with AWS X-Ray:** In addition to deploying the CloudWatch agent, it's crucial to instrument your application to emit trace data. This process can provide further insights into your workload's behavior and performance.
7. **Standardize data collection across your application:** Standardize data collection practices across your entire application. Uniformity aids in correlating and analyzing data, providing a comprehensive view of your application's behavior.
8. **Implement cross-account observability:** Enhance monitoring efficiency across multiple AWS accounts with Amazon CloudWatch cross-account observability. With this feature, you can consolidate metrics, logs, and alarms from different accounts into a single view, which simplifies

management and improves response times for identified issues across your organization's AWS environment.

9. **Analyze and act on the data:** Once data collection and normalization are in place, use Amazon CloudWatch for metrics and logs analysis, and AWS X-Ray for trace analysis. Such analysis can yield crucial insights into your workload's health, performance, and behavior, guiding your decision-making process.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS04-BP01 Define workload KPIs
- OPS04-BP03 Implement user activity telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement transaction traceability

Related documents:

- AWS Observability Best Practices
- CloudWatch User Guide
- AWS X-Ray Developer Guide
- Instrumenting distributed systems for operational visibility
- AWS Observability Skill Builder Course
- What's New with Amazon CloudWatch
- What's new with AWS X-Ray

Related videos:

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2022 - Developing an observability strategy

Related examples:

- One Observability Workshop
- AWS Solutions Library: Application Monitoring with Amazon CloudWatch

OPS04-BP03 Implement user experience telemetry

Gaining deep insights into customer experiences and interactions with your application is crucial. Real user monitoring (RUM) and synthetic transactions serve as powerful tools for this purpose. RUM provides data about real user interactions granting an unfiltered perspective of user satisfaction, while synthetic transactions simulate user interactions, helping in detecting potential issues even before they impact real users.

Desired outcome: A holistic view of the customer experience, proactive detection of issues, and optimization of user interactions to deliver seamless digital experiences.

Common anti-patterns:

- Applications without real user monitoring (RUM):
  - Delayed issue detection: Without RUM, you might not become aware of performance bottlenecks or issues until users complain. This reactive approach can lead to customer dissatisfaction.
  - Lack of user experience insights: Not using RUM means you lose out on crucial data that shows how real users interact with your application, limiting your ability to optimize the user experience.
- Applications without synthetic transactions:
  - Missed edge cases: Synthetic transactions help you test paths and functions that might not be frequently used by typical users but are critical to certain business functions. Without them, these paths could malfunction and go unnoticed.
  - Checking for issues when the application is not being used: Regular synthetic testing can simulate times when real users aren't actively interacting with your application, ensuring the system always functions correctly.

Benefits of establishing this best practice:

- Proactive issue detection: Identify and address potential issues before they impact real users.
- Optimized user experience: Continuous feedback from RUM aids in refining and enhancing the overall user experience.

- Insights on device and browser performance: Understand how your application performs across various devices and browsers, enabling further optimization.
- Validated business workflows: Regular synthetic transactions ensure that core functionalities and critical paths remain operational and efficient.
- Enhanced application performance: Leverage insights gathered from real user data to improve application responsiveness and reliability.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To leverage RUM and synthetic transactions for user activity telemetry, AWS offers services like Amazon CloudWatch RUM and Amazon CloudWatch Synthetics. Metrics, logs, and traces, coupled with user activity data, provide a comprehensive view of both the application's operational state and the user experience.

Implementation steps

1. **Deploy Amazon CloudWatch RUM:** Integrate your application with CloudWatch RUM to collect, analyze, and present real user data.
   a. Use the CloudWatch RUM JavaScript library to integrate RUM with your application.
   b. Set up dashboards to visualize and monitor real user data.
2. **Configure CloudWatch Synthetics:** Create canaries, or scripted routines, that simulate user interactions with your application.
   a. Define critical application workflows and paths.
   b. Design canaries using CloudWatch Synthetics scripts to simulate user interactions for these paths.
   c. Schedule and monitor canaries to run at specified intervals, ensuring consistent performance checks.
3. **Analyze and act on data:** Utilize data from RUM and synthetic transactions to gain insights and take corrective measures when anomalies are detected. Use CloudWatch dashboards and alarms to stay informed.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement distributed tracing

Related documents:

- Amazon CloudWatch RUM Guide
- Amazon CloudWatch Synthetics Guide

Related videos:

- Optimize applications through end user insights with Amazon CloudWatch RUM
- AWS on Air ft. Real-User Monitoring for Amazon CloudWatch

Related examples:

- One Observability Workshop
- Git Repository for Amazon CloudWatch RUM Web Client
- Using Amazon CloudWatch Synthetics to measure page load time

OPS04-BP04 Implement dependency telemetry

Dependency telemetry is essential for monitoring the health and performance of the external services and components your workload relies on. It provides valuable insights into reachability, timeouts, and other critical events related to dependencies such as DNS, databases, or third-party APIs. When you instrument your application to emit metrics, logs, and traces about these dependencies, you gain a clearer understanding of potential bottlenecks, performance issues, or failures that might impact your workload.

Desired outcome: Ensure that the dependencies your workload relies on are performing as expected, allowing you to proactively address issues and ensure optimal workload performance.

Common anti-patterns:

- **Overlooking external dependencies:** Focusing only on internal application metrics while neglecting metrics related to external dependencies.
- **Lack of proactive monitoring:** Waiting for issues to arise instead of continuously monitoring dependency health and performance.
- **Siloed monitoring:** Using multiple, disparate monitoring tools which can result in fragmented and inconsistent views of dependency health.

Benefits of establishing this best practice:

- **Improved workload reliability:** By ensuring that external dependencies are consistently available and performing optimally.
- **Faster issue detection and resolution:** Proactively identifying and addressing issues with dependencies before they impact the workload.
- **Comprehensive view:** Gaining a holistic view of both internal and external components that influence workload health.
- **Enhanced workload scalability:** By understanding the scalability limits and performance characteristics of external dependencies.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implement dependency telemetry by starting with identifying the services, infrastructure, and processes that your workload depends on. Quantify what good conditions look like when those dependencies are functioning as expected, and then determine what data will be needed to measure those. With that information you can craft dashboards and alerts that provide insights to your operations teams on the state of those dependencies. Use AWS tools to discover and quantify the impacts when dependencies cannot deliver as needed. Continually revisit your strategy to account for changes in priorities, goals, and gained insights.

Implementation steps

To implement dependency telemetry effectively:

1. **Identify external dependencies:** Collaborate with stakeholders to pinpoint the external dependencies your workload relies on. External dependencies can encompass services like

external databases, third-party APIs, network connectivity routes to other environments, and DNS services. The first step towards effective dependency telemetry is being comprehensive in understanding what those dependencies are.

2. **Develop a monitoring strategy:** Once you have a clear picture of your external dependencies, architect a monitoring strategy tailored to them. This involves understanding the criticality of each dependency, its expected behavior, and any associated service-level agreements or targets (SLA or SLTs). Set up proactive alerts to notify you of status changes or performance deviations.
3. **Use network monitoring:** Use Internet Monitor and Network Monitor, which provide comprehensive insights into global internet and network conditions. These tools help you understand and respond to outages, disruptions, or performance degradations that affect your external dependencies.
4. **Stay informed with AWS Health:** AWS Health is the authoritative source of information about the health of your AWS Cloud resources. Use AWS Health to visualize and receive notifications about any current service events and upcoming changes, such as planned lifecycle events, so you can take steps to mitigate impacts.
    a. Create purpose-fit AWS Health event notifications to e-mail and chat channels through AWS User Notifications, and integrate programatically with your monitoring and alerting tools through Amazon EventBridge or the AWS Health API.
    b. Plan and track progress on health events that require action by integrating with change management or ITSM tools (like Jira or ServiceNow) that you may already use through Amazon EventBridge or the AWS Health API.
    c. If you use AWS Organizations, enable organization view for AWS Health to aggregate AWS Health events across accounts.
5. **Instrument your application with AWS X-Ray:** AWS X-Ray provides insights into how applications and their underlying dependencies are performing. By tracing requests from start to end, you can identify bottlenecks or failures in the external services or components your application relies on.
6. **Use Amazon DevOps Guru:** This machine learning-driven service identifies operational issues, predicts when critical issues might occur, and recommends specific actions to take. It's invaluable for gaining insights into dependencies and ensuring they're not the source of operational problems.
7. **Monitor regularly:** Continually monitor metrics and logs related to external dependencies. Set up alerts for unexpected behavior or degraded performance.

8. **Validate after changes:** Whenever there's an update or change in any of the external dependencies, validate their performance and check their alignment with your application's requirements.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Define workload KPIs
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user activity telemetry
- OPS04-BP05 Implement transaction traceability
- OP08-BP04 Create actionable alerts

Related documents:

- Amazon Personal Health Dashboard User Guide
- AWS Internet Monitor User Guide
- AWS X-Ray Developer Guide
- AWS DevOps Guru User Guide

Related videos:

- Visibility into how internet issues impact app performance
- Introduction to Amazon DevOps Guru
- Manage resource lifecycle events at scale with AWS Health

Related examples:

- AWS Health Aware
- Using Tag-Based Filtering to Manage AWS Health Monitoring and Alerting at Scale

OPS04-BP05 Implement distributed tracing

Distributed tracing offers a way to monitor and visualize requests as they traverse through various components of a distributed system. By capturing trace data from multiple sources and analyzing it in a unified view, teams can better understand how requests flow, where bottlenecks exist, and where optimization efforts should focus.

Desired outcome: Achieve a holistic view of requests flowing through your distributed system, allowing for precise debugging, optimized performance, and improved user experiences.

Common anti-patterns:

- Inconsistent instrumentation: Not all services in a distributed system are instrumented for tracing.
- Ignoring latency: Only focusing on errors and not considering the latency or gradual performance degradations.

Benefits of establishing this best practice:

- Comprehensive system overview: Visualizing the entire path of requests, from entry to exit.
- Enhanced debugging: Quickly identifying where failures or performance issues occur.
- Improved user experience: Monitoring and optimizing based on actual user data, ensuring the system meets real-world demands.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Begin by identifying all of the elements of your workload that require instrumentation. Once all components are accounted for, leverage tools such as AWS X-Ray and OpenTelemetry to gather trace data for analysis with tools like X-Ray and Amazon CloudWatch ServiceLens Map. Engage in regular reviews with developers, and supplement these discussions with tools like Amazon DevOps Guru, X-Ray Analytics and X-Ray Insights to help uncover deeper findings. Establish alerts from trace data to notify when outcomes, as defined in the workload monitoring plan, are at risk.

Implementation steps

To implement distributed tracing effectively:

1. **Adopt AWS X-Ray:** Integrate X-Ray into your application to gain insights into its behavior, understand its performance, and pinpoint bottlenecks. Utilize X-Ray Insights for automatic trace analysis.
2. **Instrument your services:** Verify that every service, from an AWS Lambda function to an EC2 instance, sends trace data. The more services you instrument, the clearer the end-to-end view.
3. **Incorporate CloudWatch Real User Monitoring and synthetic monitoring:** Integrate Real User Monitoring (RUM) and synthetic monitoring with X-Ray. This allows for capturing real-world user experiences and simulating user interactions to identify potential issues.
4. **Use the CloudWatch agent:** The agent can send traces from either X-Ray or OpenTelemetry, enhancing the depth of insights obtained.
5. **Use Amazon DevOps Guru:** DevOps Guru uses data from X-Ray, CloudWatch, AWS Config, and AWS CloudTrail to provide actionable recommendations.
6. **Analyze traces:** Regularly review the trace data to discern patterns, anomalies, or bottlenecks that might impact your application's performance.
7. **Set up alerts:** Configure alarms in CloudWatch for unusual patterns or extended latencies, allowing proactive issue addressing.
8. **Continuous improvement:** Revisit your tracing strategy as services are added or modified to capture all relevant data points.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP04 Implement dependency telemetry

Related documents:

- AWS X-Ray Developer Guide
- Amazon CloudWatch agent User Guide

- Amazon DevOps Guru User Guide

Related videos:

- Use AWS X-Ray Insights
- AWS on Air ft. Observability: Amazon CloudWatch and AWS X-Ray

Related examples:

- Instrumenting your application for AWS X-Ray

OPS 5. How do you reduce defects, ease remediation, and improve flow into production?

Adopt approaches that improve flow of changes into production, that activate refactoring, fast feedback on quality, and bug fixing. These accelerate beneficial changes entering production, limit issues deployed, and achieve rapid identification and remediation of issues introduced through deployment activities.

Best practices

- OPS05-BP01 Use version control
- OPS05-BP02 Test and validate changes
- OPS05-BP03 Use configuration management systems
- OPS05-BP04 Use build and deployment management systems
- OPS05-BP05 Perform patch management
- OPS05-BP06 Share design standards
- OPS05-BP07 Implement practices to improve code quality
- OPS05-BP08 Use multiple environments
- OPS05-BP09 Make frequent, small, reversible changes
- OPS05-BP10 Fully automate integration and deployment

OPS05-BP01 Use version control

Use version control to activate tracking of changes and releases.

Many AWS services offer version control capabilities. Use a revision or source control system such as Git to manage code and other artifacts such as version-controlled AWS CloudFormation templates of your infrastructure.

Desired outcome: Your teams collaborate on code. When merged, the code is consistent and no changes are lost. Errors are easily reverted through correct versioning.

Common anti-patterns:

- You have been developing and storing your code on your workstation. You have had an unrecoverable storage failure on the workstation and your code is lost.
- After overwriting the existing code with your changes, you restart your application and it is no longer operable. You are unable to revert the change.
- You have a write lock on a report file that someone else needs to edit. They contact you asking that you stop work on it so that they can complete their tasks.
- Your research team has been working on a detailed analysis that shapes your future work. Someone has accidentally saved their shopping list over the final report. You are unable to revert the change and have to recreate the report.

Benefits of establishing this best practice: By using version control capabilities you can easily revert to known good states and previous versions, and limit the risk of assets being lost.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Maintain assets in version controlled repositories. Doing so supports tracking changes, deploying new versions, detecting changes to existing versions, and reverting to prior versions (for example, rolling back to a known good state in the event of a failure). Integrate the version control capabilities of your configuration management systems into your procedures.

Resources

Related best practices:

- OPS05-BP04 Use build and deployment management systems

Related videos:

- AWS re:Invent 2023 - How Lockheed Martin builds software faster, powered by DevSecOps
- AWS re:Invent 2023 - How GitHub operationalizes AI for team collaboration and productivity

OPS05-BP02 Test and validate changes

Every change deployed must be tested to avoid errors in production. This best practice is focused on testing changes from version control to artifact build. Besides application code changes, testing should include infrastructure, configuration, security controls, and operations procedures. Testing takes many forms, from unit tests to software component analysis (SCA). Move tests further to the left in the software integration and delivery process results in higher certainty of artifact quality.

Your organization must develop testing standards for all software artifacts. Automated tests reduce toil and avoid manual test errors. Manual tests may be necessary in some cases. Developers must have access to automated test results to create feedback loops that improve software quality.

Desired outcome: Your software changes are tested before they are delivered. Developers have access to test results and validations. Your organization has a testing standard that applies to all software changes.

Common anti-patterns:

- You deploy a new software change without any tests. It fails to run in production, which leads to an outage.
- New security groups are deployed with AWS CloudFormation without being tested in a pre-production environment. The security groups make your app unreachable for your customers.
- A method is modified but there are no unit tests. The software fails when it is deployed to production.

Benefits of establishing this best practice: Change fail rate of software deployments are reduced. Software quality is improved. Developers have increased awareness on the viability of their code. Security policies can be rolled out with confidence to support organization's compliance. Infrastructure changes such as automatic scaling policy updates are tested in advance to meet traffic needs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Testing is done on all changes, from application code to infrastructure, as part of your continuous integration practice. Test results are published so that developers have fast feedback. Your organization has a testing standard that all changes must pass.

Use the power of generative AI with Amazon Q Developer to improve developer productivity and code quality. Amazon Q Developer includes generation of code suggestions (based on large language models), production of unit tests (including boundary conditions), and code security enhancements through detection and remediation of security vulnerabilities.

Customer example

As part of their continuous integration pipeline, AnyCompany Retail conducts several types of tests on all software artifacts. They practice test driven development so all software has unit tests. Once the artifact is built, they run end-to-end tests. After this first round of tests is complete, they run a static application security scan, which looks for known vulnerabilities. Developers receive messages as each testing gate is passed. Once all tests are complete, the software artifact is stored in an artifact repository.

Implementation steps

1. Work with stakeholders in your organization to develop a testing standard for software artifacts. What standard tests should all artifacts pass? Are there compliance or governance requirements that must be included in the test coverage? Do you need to conduct code quality tests? When tests complete, who needs to know?
    1. The AWS Deployment Pipeline Reference Architecture contains an authoritative list of types of tests that can be conducted on software artifacts as part of an integration pipeline.
2. Instrument your application with the necessary tests based on your software testing standard. Each set of tests should complete in under ten minutes. Tests should run as part of an integration pipeline.
    a. Use Amazon Q Developer, a generative AI tool that can help create unit test cases (including boundary conditions), generate functions using code and comments, and implement well-known algorithms.
    b. Use Amazon CodeGuru Reviewer to test your application code for defects.
    c. You can use AWS CodeBuild to conduct tests on software artifacts.
    d. AWS CodePipeline can orchestrate your software tests into a pipeline.

Resources

Related best practices:

- OPS05-BP01 Use version control
- OPS05-BP06 Share design standards
- OPS05-BP07 Implement practices to improve code quality
- OPS05-BP10 Fully automate integration and deployment

Related documents:

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Automated AWS CloudFormation Testing Pipeline with TaskCat and CodePipeline
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST, and DAST tools
- Getting started with testing serverless applications
- My CI/CD pipeline is my release captain
- Practicing Continuous Integration and Continuous Delivery on AWS Whitepaper

Related videos:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)

- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Testable infrastructure: Integration testing on AWS
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development
- Testing Your Infrastructure as Code with AWS CDK

Related resources:

- AWS Deployment Pipeline Reference Architecture - Application
- AWS Kubernetes DevSecOps Pipeline
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild
- Use Serverspec for test-driven development of infrastructure code

Related services:

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- AWS CodeBuild
- AWS CodePipeline

OPS05-BP03 Use configuration management systems

Use configuration management systems to make and track configuration changes. These systems reduce errors caused by manual processes and reduce the level of effort to deploy changes.

Static configuration management sets values when initializing a resource that are expected to remain consistent throughout the resource's lifetime. Dynamic configuration management sets values at initialization that can or are expected to change during the lifetime of a resource. For example, you could set a feature toggle to activate functionality in your code through a configuration change, or change the level of log detail during an incident.

Configurations should be deployed in a known and consistent state. You should use automated inspection to continually monitor resource configurations across environments and regions. These controls should be defined as code and management automated to ensure rules are consistently appplied across environments. Changes to configurations should be updated through agreed change control procedures and applied consistently, honoring version control. Application

configuration should be managed independently of application and infrastructure code. This allows for consistent deployment across multiple environments. Configuration changes do not result in rebuilding or redeploying the application.

Desired outcome: You configure, validate, and deploy as part of your continuous integration, continuous delivery (CI/CD) pipeline. You monitor to validate configurations are correct. This minimizes any impact to end users and customers.

Common anti-patterns:

- You manually update the web server configuration across your fleet and a number of servers become unresponsive due to update errors.
- You manually update your application server fleet over the course of many hours. The inconsistency in configuration during the change causes unexpected behaviors.
- Someone has updated your security groups and your web servers are no longer accessible. Without knowledge of what was changed you spend significant time investigating the issue extending your time to recovery.
- You push a pre-production configuration into production through CI/CD without validation. You expose users and customers to incorrect data and services.

Benefits of establishing this best practice: Adopting configuration management systems reduces the level of effort to make and track changes, and the frequency of errors caused by manual procedures. Configuration management systems provide assurances with regards to governance, compliance, and regulatory requirements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Configuration management systems are used to track and implement changes to application and environment configurations. Configuration management systems are also used to reduce errors caused by manual processes, make configuration changes repeatable and auditable, and reduce the level of effort.

On AWS, you can use AWS Config to continually monitor your AWS resource configurations across accounts and Regions. It helps you to track their configuration history, understand how a configuration change would affect other resources, and audit them against expected or desired configurations using AWS Config Rules and AWS Config Conformance Packs.

For dynamic configurations in your applications running on Amazon EC2 instances, AWS Lambda, containers, mobile applications, or IoT devices, you can use AWS AppConfig to configure, validate, deploy, and monitor them across your environments.

Implementation steps

1. Identify configuration owners.
   a. Make configurations owners aware of any compliance, governance, or regulatory needs.
2. Identify configuration items and deliverables.
   a. Configuration items are all application and environmental configurations affected by a deployment within your CI/CD pipeline.
   b. Deliverables include success criteria, validation, and what to monitor.
3. Select tools for configuration management based on your business requirements and delivery pipeline.
4. Consider weighted deployments such as canary deployments for significant configuration changes to minimize the impact of incorrect configurations.
5. Integrate your configuration management into your CI/CD pipeline.
6. Validate all changes pushed.

Resources

Related best practices:

- OPS06-BP01 Plan for unsuccessful changes
- OPS06-BP02 Test deployments
- OPS06-BP03 Employ safe deployment strategies
- OPS06-BP04 Automate testing and rollback

Related documents:

- AWS Control Tower
- AWS Landing Zone Accelerator
- AWS Config
- What is AWS Config?

- AWS AppConfig
- What is AWS CloudFormation?
- AWS Developer Tools
- AWS CodeBuild
- AWS CodePipeline
- AWS CodeDeploy

Related videos:

- AWS re:Invent 2022 - Proactive governance and compliance for AWS workloads
- AWS re:Invent 2020: Achieve compliance as code using AWS Config
- Manage and Deploy Application Configurations with AWS AppConfig

OPS05-BP04 Use build and deployment management systems

Use build and deployment management systems. These systems reduce errors caused by manual processes and reduce the level of effort to deploy changes.

In AWS, you can build continuous integration/continuous deployment (CI/CD) pipelines using services such as AWS Developer Tools (for example, AWS CodeBuild, AWS CodePipeline, and AWS CodeDeploy).

Desired outcome: Your build and deployment management systems support your organization's continuous integration continuous delivery (CI/CD) system that provide capabilities for automating safe rollouts with the correct configurations.

Common anti-patterns:

- After compiling your code on your development system, you copy the executable onto your production systems and it fails to start. The local log files indicates that it has failed due to missing dependencies.
- You successfully build your application with new features in your development environment and provide the code to quality assurance (QA). It fails QA because it is missing static assets.
- On Friday, after much effort, you successfully built your application manually in your development environment including your newly coded features. On Monday, you are unable to repeat the steps that allowed you to successfully build your application.

- You perform the tests you have created for your new release. Then you spend the next week setting up a test environment and performing all the existing integration tests followed by the performance tests. The new code has an unacceptable performance impact and must be redeveloped and then retested.

Benefits of establishing this best practice: By providing mechanisms to manage build and deployment activities you reduce the level of effort to perform repetitive tasks, free your team members to focus on their high value creative tasks, and limit the introduction of error from manual procedures.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Build and deployment management systems are used to track and implement change, reduce errors caused by manual processes, and reduce the level of effort required for safe deployments. Fully automate the integration and deployment pipeline from code check-in through build, testing, deployment, and validation. This reduces lead time, decreases cost, encourages increased frequency of change, reduces the level of effort, and increases collaboration.

Implementation steps

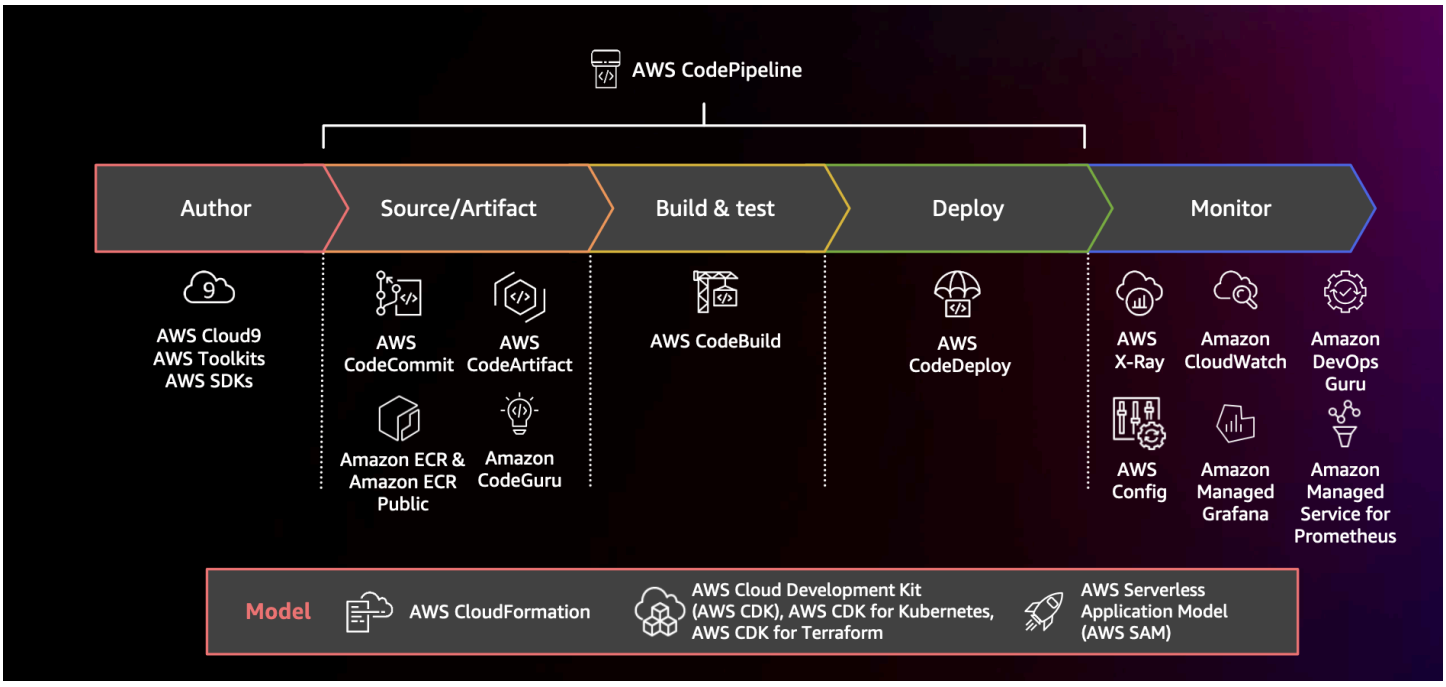


Diagram showing a CI/CD pipeline using AWS CodePipeline and related services

1. Use a version control system to store and manage assets (such as documents, source code, and binary files).
2. Use CodeBuild to compile your source code, runs unit tests, and produces artifacts that are ready to deploy.
3. Use CodeDeploy as a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, serverless AWS Lambda functions, or Amazon ECS.
4. Monitor your deployments.

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

- AWS Developer Tools
- What is AWS CodeBuild?
- AWS CodeBuild
- What is AWS CodeDeploy?

Related videos:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

OPS05-BP05 Perform patch management

Perform patch management to gain features, address issues, and remain compliant with governance. Automate patch management to reduce errors caused by manual processes, scale, and reduce the level of effort to patch.

Patch and vulnerability management are part of your benefit and risk management activities. It is preferable to have immutable infrastructures and deploy workloads in verified known good states. Where that is not viable, patching in place is the remaining option.

AWS Health is the authoritative source of information about planned lifecycle events and other action-required events that affect the health of your AWS Cloud resources. You should be aware of

upcoming changes and updates that should be performed. Major planned lifecycle events are sent at least six months in advance.

Amazon EC2 Image Builder provides pipelines to update machine images. As a part of patch management, consider Amazon Machine Images (AMIs) using an AMI image pipeline or container images with a Docker image pipeline, while AWS Lambda provides patterns for custom runtimes and additional libraries to remove vulnerabilities.

You should manage updates to Amazon Machine Images for Linux or Windows Server images using Amazon EC2 Image Builder. You can use Amazon Elastic Container Registry (Amazon ECR) with your existing pipeline to manage Amazon ECS images and manage Amazon EKS images. Lambda includes version management features.

Patching should not be performed on production systems without first testing in a safe environment. Patches should only be applied if they support an operational or business outcome. On AWS, you can use AWS Systems Manager Patch Manager to automate the process of patching managed systems and schedule the activity using Systems Manager Maintenance Windows.

Desired outcome: Your AMI and container images are patched, up-to-date, and ready for launch. You are able to track the status of all deployed images and know patch compliance. You are able to report on current status and have a process to meet your compliance needs.

Common anti-patterns:

- You are given a mandate to apply all new security patches within two hours resulting in multiple outages due to application incompatibility with patches.
- An unpatched library results in unintended consequences as unknown parties use vulnerabilities within it to access your workload.
- You patch the developer environments automatically without notifying the developers. You receive multiple complaints from the developers that their environment cease to operate as expected.
- You have not patched the commercial off-the-shelf software on a persistent instance. When you have an issue with the software and contact the vendor, they notify you that version is not supported and you have to patch to a specific level to receive any assistance.
- A recently released patch for the encryption software you used has significant performance improvements. Your unpatched system has performance issues that remain in place as a result of not patching.

- You are notified of a zero-day vulnerability requiring an emergency fix and you have to patch all your environments manually.
- You are not aware of critical actions needed to maintain your resources, such as mandatory version updates because you do not review upcoming planned lifecycle events and other information. You lose critical time for planning and execution, resulting in emergency changes for your teams and potential impact or unexpected downtime.

Benefits of establishing this best practice: By establishing a patch management process, including your criteria for patching and methodology for distribution across your environments, you can scale and report on patch levels. This provides assurances around security patching and ensure clear visibility on the status of known fixes being in place. This encourages adoption of desired features and capabilities, the rapid removal of issues, and sustained compliance with governance. Implement patch management systems and automation to reduce the level of effort to deploy patches and limit errors caused by manual processes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Patch systems to remediate issues, to gain desired features or capabilities, and to remain compliant with governance policy and vendor support requirements. In immutable systems, deploy with the appropriate patch set to achieve the desired result. Automate the patch management mechanism to reduce the elapsed time to patch, to avoid errors caused by manual processes, and lower the level of effort to patch.

Implementation steps

For Amazon EC2 Image Builder:

1. Using Amazon EC2 Image Builder, specify pipeline details:
    a. Create an image pipeline and name it
    b. Define pipeline schedule and time zone
    c. Configure any dependencies
2. Choose a recipe:
    a. Select existing recipe or create a new one
    b. Select image type
    c. Name and version your recipe

   d. Select your base image
   e. Add build components and add to target registry
3. Optional - define your infrastructure configuration.
4. Optional - define configuration settings.
5. Review settings.
6. Maintain recipe hygiene regularly.

For Systems Manager Patch Manager:

1. Create a patch baseline.
2. Select a patching operations method.
3. Enable compliance reporting and scanning.

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

- What is Amazon EC2 Image Builder
- Create an image pipeline using the Amazon EC2 Image Builder
- Create a container image pipeline
- AWS Systems Manager Patch Manager
- Working with Patch Manager
- Working with patch compliance reports
- AWS Developer Tools

Related videos:

- CI/CD for Serverless Applications on AWS
- Design with Ops in Mind

Related examples:

- AWS Systems Manager Patch Manager tutorials

OPS05-BP06 Share design standards

Share best practices across teams to increase awareness and maximize the benefits of development efforts. Document them and keep them up to date as your architecture evolves. If shared standards are enforced in your organization, it's critical that mechanisms exist to request additions, changes, and exceptions to standards. Without this option, standards become a constraint on innovation.

Desired outcome: Design standards are shared across teams in your organizations. They are documented and kept up-to-date as best practices evolve.

Common anti-patterns:

- Two development teams have each created a user authentication service. Your users must maintain a separate set of credentials for each part of the system they want to access.
- Each team manages their own infrastructure. A new compliance requirement forces a change to your infrastructure and each team implements it in a different way.

Benefits of establishing this best practice: Using shared standards supports the adoption of best practices and maximizes the benefits of development efforts. Documenting and updating design standards keeps your organization up-to-date with best practices and security and compliance requirements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Share existing best practices, design standards, checklists, operating procedures, guidance, and governance requirements across teams. Have procedures to request changes, additions, and exceptions to design standards to support improvement and innovation. Make teams are aware of published content. Have a mechanism to keep design standards up-to-date as new best practices emerge.

Customer example

AnyCompany Retail has a cross-functional architecture team that creates software architecture patterns. This team builds the architecture with compliance and governance built in. Teams that

adopt these shared standards get the benefits of having compliance and governance built in. They can quickly build on top of the design standard. The architecture team meets quarterly to evaluate architecture patterns and update them if necessary.

Implementation steps

1. Identify a cross-functional team that owns developing and updating design standards. This team should work with stakeholders across your organization to develop design standards, operating procedures, checklists, guidance, and governance requirements. Document the design standards and share them within your organization.
    a. AWS Service Catalog can be used to create portfolios representing design standards using infrastructure as code. You can share portfolios across accounts.
2. Have a mechanism in place to keep design standards up-to-date as new best practices are identified.
3. If design standards are centrally enforced, have a process to request changes, updates, and exemptions.

Level of effort for the implementation plan: Medium. Developing a process to create and share design standards can take coordination and cooperation with stakeholders across your organization.

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements - Governance requirements influence design standards.
- OPS01-BP04 Evaluate compliance requirements - Compliance is a vital input in creating design standards.
- OPS07-BP02 Ensure a consistent review of operational readiness - Operational readiness checklists are a mechanism to implement design standards when designing your workload.
- OPS11-BP01 Have a process for continuous improvement - Updating design standards is a part of continuous improvement.
- OPS11-BP04 Perform knowledge management - As part of your knowledge management practice, document and share design standards.

Related documents:

- Automate AWS Backups with AWS Service Catalog
- AWS Service Catalog Account Factory-Enhanced
- How Expedia Group built Database as a Service (DBaaS) offering using AWS Service Catalog
- Maintain visibility over the use of cloud architecture patterns
- Simplify sharing your AWS Service Catalog portfolios in an AWS Organizations setup

Related videos:

- AWS Service Catalog – Getting Started
- AWS re:Invent 2020: Manage your AWS Service Catalog portfolios like an expert

Related examples:

- AWS Service Catalog Reference Architecture
- AWS Service Catalog Workshop

Related services:

- AWS Service Catalog

OPS05-BP07 Implement practices to improve code quality

Implement practices to improve code quality and minimize defects. Some examples include test-driven development, code reviews, standards adoption, and pair programming. Incorporate these practices into your continuous integration and delivery process.

Desired outcome: Your organization uses best practices like code reviews or pair programming to improve code quality. Developers and operators adopt code quality best practices as part of the software development lifecycle.

Common anti-patterns:

- You commit code to the main branch of your application without a code review. The change automatically deploys to production and causes an outage.

- A new application is developed without any unit, end-to-end, or integration tests. There is no way to test the application before deployment.
- Your teams make manual changes in production to address defects. Changes do not go through testing or code reviews and are not captured or logged through continuous integration and delivery processes.

Benefits of establishing this best practice: By adopting practices to improve code quality, you can help minimize issues introduced to production. Code quality best practices include pair programming, code reviews, and implementation of AI productivity tools.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement practices to improve code quality to minimize defects before they are deployed. Use practices like test-driven development, code reviews, and pair programming to increase the quality of your development.

Use the power of generative AI with Amazon Q Developer to improve developer productivity and code quality. Amazon Q Developer includes generation of code suggestions (based on large language models), production of unit tests (including boundary conditions), and code security enhancements through detection and remediation of security vulnerabilities.

Customer example

AnyCompany Retail adopts several practices to improve code quality. They have adopted test-driven development as the standard for writing applications. For some new features, they will have developers pair program together during a sprint. Every pull request goes through a code review by a senior developer before being integrated and deployed.

Implementation steps

1. Adopt code quality practices like test-driven development, code reviews, and pair programming into your continuous integration and delivery process. Use these techniques to improve software quality.
   a. Use Amazon Q Developer, a generative AI tool that can help create unit test cases (including boundary conditions), generate functions using code and comments, implement well-known algorithms, detect security policy violations and vulnerabilities in your code, detect secrets,

scan infrastructure as code (IaC), document code, and learn third-party code libraries more quickly.

b. Amazon CodeGuru Reviewer can provide programming recommendations for Java and Python code using machine learning.

Level of effort for the implementation plan: Medium. There are many ways of implementing this best practice, but getting organizational adoption may be challenging.

Resources

Related best practices:

- OPS05-BP02 Test and validate changes
- OPS05-BP06 Share design standards

Related documents:

- Adopt a test-driven development approach
- Accelerate your Software Development Lifecycle with Amazon Q
- Amazon Q Developer, now generally available, includes previews of new capabilities to reimagine developer experience
- The Ultimate Cheat Sheet for Using Amazon Q Developer in Your IDE
- Shift-Left Workload, leveraging AI for Test Creation
- Amazon Q Developer Center
- 10 ways to build applications faster with Amazon CodeWhisperer
- Looking beyond code coverage with Amazon CodeWhisperer
- Best Practices for Prompt Engineering with Amazon CodeWhisperer
- Agile Software Guide
- My CI/CD pipeline is my release captain
- Automate code reviews with Amazon CodeGuru Reviewer
- Adopt a test-driven development approach
- How DevFactory builds better applications with Amazon CodeGuru
- On Pair Programming

- RENGA Inc. automates code reviews with Amazon CodeGuru
- The Art of Agile Development: Test-Driven Development
- Why code reviews matter (and actually save time!)

Related videos:

- Implement an API with Amazon Q Developer Agent for Software Development
- Installing, Configuring, & Using Amazon Q Developer with JetBrains IDEs (How-to)
- Mastering the art of Amazon CodeWhisperer - YouTube playlist
- AWS re:Invent 2020: Continuous improvement of code quality with Amazon CodeGuru
- AWS Summit ANZ 2021 - Driving a test-first strategy with CDK and test driven development

Related services:

- Amazon Q Developer
- Amazon CodeGuru Reviewer
- Amazon CodeGuru Profiler

OPS05-BP08 Use multiple environments

Use multiple environments to experiment, develop, and test your workload. Use increasing levels of controls as environments approach production to gain confidence your workload operates as intended when deployed.

Desired outcome: You have multiple environments that reflect your compliance and governance needs. You test and promote code through environments on your path to production.

1. Your organization does this through the establishment of a landing zone, which provides governance, controls, account automations, networking, security, and operational observability. Manage these landing zone capabilities by using multiple environments. A common example is a sandbox organization for developing and testing changes to an AWS Control Tower-based landing zone, which includes AWS IAM Identity Center and policies such as service control policies (SCPs). All of these elements can significantly impact the access to and operation of AWS accounts within the landing zone.
2. In addition to these services, your teams extend the landing zones capabilites with solutions published by AWS and AWS partners or as custom solutions developed within your organization.

Examples of solutions published by AWS include Customizations for AWS Control Tower (CfCT) and AWS Control Tower Account Factory for Terraform (AFT).

3. Your organization applies the same principles of testing, promoting code, and policy changes for the landing zone through environments on your path to production. This strategy provides a stable and secure landing zone environment for your application and workload teams.

Common anti-patterns:

- You are performing development in a shared development environment and another developer overwrites your code changes.
- The restrictive security controls on your shared development environment are preventing you from experimenting with new services and features.
- You perform load testing on your production systems and cause an outage for your users.
- A critical error resulting in data loss has occurred in production. In your production environment, you attempt to recreate the conditions that lead to the data loss so that you can identify how it happened and prevent it from happening again. To prevent further data loss during testing, you are forced to make the application unavailable to your users.
- You are operating a multi-tenant service and are unable to support a customer request for a dedicated environment.
- You may not always test, but when you do, you test in your production environment.
- You believe that the simplicity of a single environment overrides the scope of impact of changes within the environment.
- You upgrade a key landing zone capability, but the change impairs your team's ability to vend accounts for either new projects or your existing workloads.
- You apply new controls to your AWS accounts, but the change impacts your workload team's ability to deploy changes within their AWS accounts.

Benefits of establishing this best practice: When you deploy multiple environments, you can support multiple simultaneous development, testing, and production environments without creating conflicts between developers or user communities. For complex capabilities such as landing zones, it significantly reduces the risk of changes, simplifies the improvement process, and reduces the risk of critical updates to the environment. Organizations that use landing zones naturally benefit from multi-accounts in their AWS environment, with account structure,

governance, network, and security configurations. Over time, as your organization grows, the landing zone can evolve to secure and organize your workloads and resources.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use multiple environments and provide developers sandbox environments with minimized controls to aid in experimentation. Provide individual development environments to help work in parallel, increasing development agility. Implement more rigorous controls in the environments approaching production to allow developers to innovate. Use infrastructure as code and configuration management systems to deploy environments that are configured consistent with the controls present in production to ensure systems operate as expected when deployed. When environments are not in use, turn them off to avoid costs associated with idle resources (for example, development systems on evenings and weekends). Deploy production equivalent environments when load testing to improve valid results.

Teams such as platform engineering, networking, and security operations often manage capabilies at the organization level with distinct requirements. A separation of accounts alone is insufficient to provide and maintain separate environments for experimentation, development, and testing. In such cases, create separate instances of AWS Organizations.

Resources

Related documents:

- Instance Scheduler on AWS
- What is AWS CloudFormation?
- Organizing Your AWS Environment Using Multiple Accounts - Multiple organizations - Test changes to your overall AWS environment
- AWS Control Tower Guide

OPS05-BP09 Make frequent, small, reversible changes

Frequent, small, and reversible changes reduce the scope and impact of a change. When used in conjunction with change management systems, configuration management systems, and build and delivery systems frequent, small, and reversible changes reduce the scope and impact of a change. This results in more effective troubleshooting and faster remediation with the option to roll back changes.

Common anti-patterns:

- You deploy a new version of your application quarterly with a change window that means a core service is turned off.
- You frequently make changes to your database schema without tracking changes in your management systems.
- You perform manual in-place updates, overwriting existing installations and configurations, and have no clear roll-back plan.

Benefits of establishing this best practice: Development efforts are faster by deploying small changes frequently. When the changes are small, it is much easier to identify if they have unintended consequences, and they are easier to reverse. When the changes are reversible, there is less risk to implementing the change, as recovery is simplified. The change process has a reduced risk and the impact of a failed change is reduced.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use frequent, small, and reversible changes to reduce the scope and impact of a change. This eases troubleshooting, helps with faster remediation, and provides the option to roll back a change. It also increases the rate at which you can deliver value to the business.

Resources

Related best practices:

- OPS05-BP03 Use configuration management systems
- OPS05-BP04 Use build and deployment management systems
- OPS06-BP04 Automate testing and rollback

Related documents:

- Implementing Microservices on AWS
- Microservices - Observability

OPS05-BP10 Fully automate integration and deployment

Automate build, deployment, and testing of the workload. This reduces errors caused by manual processes and reduces the effort to deploy changes.

Apply metadata using Resource Tags and AWS Resource Groups following a consistent tagging strategy to aid in identification of your resources. Tag your resources for organization, cost accounting, access controls, and targeting the run of automated operations activities.

Desired outcome: Developers use tools to deliver code and promote through to production. Developers do not have to log into the AWS Management Console to deliver updates. There is a full audit trail of change and configuration, meeting the needs of governance and compliance. Processes are repeatable and are standardized across teams. Developers are free to focus on development and code pushes, increasing productivity.

Common anti-patterns:

- On Friday, you finish authoring the new code for your feature branch. On Monday, after running your code quality test scripts and each of your unit tests scripts, you check in your code for the next scheduled release.
- You are assigned to code a fix for a critical issue impacting a large number of customers in production. After testing the fix, you commit your code and email change management to request approval to deploy it to production.
- As a developer, you log into the AWS Management Console to create a new development environment using non-standard methods and systems.

Benefits of establishing this best practice: By implementing automated build and deployment management systems, you reduce errors caused by manual processes and reduce the effort to deploy changes helping your team members to focus on delivering business value. You increase the speed of delivery as you promote through to production.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

You use build and deployment management systems to track and implement change, to reduce errors caused by manual processes, and reduce the level of effort. Fully automate the integration and deployment pipeline from code check-in through build, testing, deployment, and validation. This reduces lead time, encourages increased frequency of change, reduces the level of effort,

increases the speed to market, results in increased productivity, and increases the security of your code as you promote through to production.

Resources

Related best practices:

- OPS05-BP03 Use configuration management systems
- OPS05-BP04 Use build and deployment management systems

Related documents:

- What is AWS CodeBuild?
- What is AWS CodeDeploy?

Related videos:

- AWS re:Invent 2022 - AWS Well-Architected best practices for DevOps on AWS

OPS 6. How do you mitigate deployment risks?

Adopt approaches that provide fast feedback on quality and achieve rapid recovery from changes that do not have desired outcomes. Using these practices mitigates the impact of issues introduced through the deployment of changes.

Best practices

- OPS06-BP01 Plan for unsuccessful changes
- OPS06-BP02 Test deployments
- OPS06-BP03 Employ safe deployment strategies
- OPS06-BP04 Automate testing and rollback

OPS06-BP01 Plan for unsuccessful changes

Plan to revert to a known good state, or remediate in the production environment if the deployment causes an undesired outcome. Having a policy to establish such a plan helps all teams develop strategies to recover from failed changes. Some example strategies are deployment and rollback steps, change policies, feature flags, traffic isolation, and traffic shifting. A single release

may include multiple related component changes. The strategy should provide the ability to withstand or recover from a failure of any component change.

Desired outcome: You have prepared a detailed recovery plan for your change in the event it is unsuccessful. In addition, you have reduced the size of your release to minimize the potential impact on other workload components. As a result, you have reduced your business impact by shortening the potential downtime caused by a failed change and increased the flexibility and efficiency of recovery times.

Common anti-patterns:

- You performed a deployment and your application has become unstable but there appear to be active users on the system. You have to decide whether to rollback the change and impact the active users or wait to rollback the change knowing the users may be impacted regardless.
- After making a routine change, your new environments are accessible, but one of your subnets has become unreachable. You have to decide whether to rollback everything or try to fix the inaccessible subnet. While you are making that determination, the subnet remains unreachable.
- Your systems are not architected in a way that allows them to be updated with smaller releases. As a result, you have difficulty in reversing those bulk changes during a failed deployment.
- You do not use infrastructure as code (IaC) and you made manual updates to your infrastructure that resulted in an undesired configuration. You are unable to effectively track and revert the manual changes.
- Because you have not measured increased frequency of your deployments, your team is not incentivized to reduce the size of their changes and improve their rollback plans for each change, leading to more risk and increased failure rates.
- You do not measure the total duration of an outage caused by unsuccessful changes. Your team is unable to prioritize and improve its deployment process and recovery plan effectiveness.

Benefits of establishing this best practice: Having a plan to recover from unsuccessful changes minimizes the mean time to recover (MTTR) and reduces your business impact.

Level of risk exposed if this best practice is not established: High

Implementation guidance

A consistent, documented policy and practice adopted by release teams allows an organization to plan what should happen if unsuccessful changes occur. The policy should allow for fixing

forward in specific circumstances. In either situation, a fix forward or rollback plan should be well documented and tested before deployment to live production so that the time it takes to revert a change is minimized.

Implementation steps

1. Document the policies that require teams to have effective plans to reverse changes within a specified period.
   a. Policies should specify when a fix-forward situation is allowed.
   b. Require a documented rollback plan to be accessible by all involved.
   c. Specify the requirements to rollback (for example, when it is found that unauthorized changes have been deployed).
2. Analyze the level of impact of all changes related to each component of a workload.
   a. Allow repeatable changes to be standardized, templated, and preauthorized if they follow a consistent workflow that enforces change policies.
   b. Reduce the potential impact of any change by making the size of the change smaller so recovery takes less time and causes less business impact.
   c. Ensure rollback procedures revert code to the known good state to avoid incidents where possible.
3. Integrate tools and workflows to enforce your policies programatically.
4. Make data about changes visible to other workload owners to improve the speed of diagnosis of any failed change that cannot be rolled back.
   a. Measure success of this practice using visible change data and identify iterative improvements.
5. Use monitoring tools to verify the success or failure of a deployment to speed up decision-making on rolling back.
6. Measure your duration of outage during an unsuccessful change to continually improve your recovery plans.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS06-BP04 Automate testing and rollback

Related documents:

- AWS Builders Library | Ensuring Rollback Safety During Deployments
- AWS Whitepaper | Change Management in the Cloud

Related videos:

- re:Invent 2019 | Amazon's approach to high-availability deployment

OPS06-BP02 Test deployments

Test release procedures in pre-production by using the same deployment configuration, security controls, steps, and procedures as in production. Validate that all deployed steps are completed as expected, such as inspecting files, configurations, and services. Further test all changes with functional, integration, and load tests, along with any monitoring such as health checks. By doing these tests, you can identify deployment issues early with an opportunity to plan and mitigate them prior to production.

You can create temporary parallel environments for testing every change. Automate the deployment of the test environments using infrastructure as code (IaC) to help reduce amount of work involved and ensure stability, consistency, and faster feature delivery.

Desired outcome: Your organization adopts a test-driven development culture that includes testing deployments. This ensures teams are focused on delivering business value rather than managing releases. Teams are engaged early upon identification of deployment risks to determine the appropriate course of mitigation.

Common anti-patterns:

- During production releases, untested deployments cause frequent issues that require troubleshooting and escalation.
- Your release contains infrastructure as code (IaC) that updates existing resources. You are unsure if the IaC runs successfully or causes impact to the resources.
- You deploy a new feature to your application. It doesn't work as intended and there is no visibility until it gets reported by impacted users.
- You update your certificates. You accidentally install the certificates to the wrong components, which goes undetected and impacts website visitors because a secure connection to the website can't be established.

Benefits of establishing this best practice: Extensive testing in pre-production of deployment procedures, and the changes introduced by them, minimizes the potential impact to production caused by the deployments steps. This increases confidence during production release and minimizes operational support without slowing down velocity of the changes being delivered.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Testing your deployment process is as important as testing the changes that result from your deployment. This can be achieved by testing your deployment steps in a pre-production environment that mirrors production as closely as possible. Common issues, such as incomplete or incorrect deployment steps, or misconfigurations, can be caught as a result before going to production. In addition, you can test your recovery steps.

Customer example

As part of their continuous integration and continuous delivery (CI/CD) pipeline, AnyCompany Retail performs the defined steps needed to release infrastructure and software updates for its customers in a production-like environment. The pipeline is comprised of pre-checks to detect drift (detecting changes to resources performed outside of your IaC) in resources prior to deployment, as well as validate actions that the IaC takes upon its initiation. It validates deployment steps, like verifying that certain files and configurations are in place and services are in running states and are responding correctly to health checks on local host before re-registering with the load balancer. Additionally, all changes flag a number of automated tests, such as functional, security, regression, integration, and load tests.

Implementation steps

1. Perform pre-install checks to mirror the pre-production environment to production.
   a. Use drift detection to detect when resources have been changed outside of CloudFormation.
   b. Use change sets to validate that the intent of a stack update matches the actions that CloudFormation takes when the change set is initiated.
2. This triggers a manual approval step in AWS CodePipeline to authorize the deployment to the pre-production environment.
3. Use deployment configurations such as AWS CodeDeploy AppSpec files to define deployment and validation steps.
4. Where applicable, integrate AWS CodeDeploy with other AWS services or integrate AWS CodeDeploy with partner product and services.

5. Monitor deployments using Amazon CloudWatch, AWS CloudTrail, and Amazon SNS event notifications.
6. Perform post-deployment automated testing, including functional, security, regression, integration, and load testing.
7. Troubleshoot deployment issues.
8. Successful validation of preceding steps should initiate a manual approval workflow to authorize deployment to production.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS05-BP02 Test and validate changes

Related documents:

- AWS Builders' Library | Automating safe, hands-off deployments | Test Deployments
- AWS Whitepaper | Practicing Continuous Integration and Continuous Delivery on AWS
- The Story of Apollo - Amazon's Deployment Engine
- How to test and debug AWS CodeDeploy locally before you ship your code
- Integrating Network Connectivity Testing with Infrastructure Deployment

Related videos:

- re:Invent 2020 | Testing software and systems at Amazon

Related examples:

- Tutorial | Deploy and Amazon ECS service with a validation test

OPS06-BP03 Employ safe deployment strategies

Safe production roll-outs control the flow of beneficial changes with an aim to minimize any perceived impact for customers from those changes. The safety controls provide inspection

mechanisms to validate desired outcomes and limit the scope of impact from any defects introduced by the changes or from deployment failures. Safe roll-outs may include strategies such as feature-flags, one-box, rolling (canary releases), immutable, traffic splitting, and blue/green deployments.

Desired outcome: Your organization uses a continuous integration continuous delivery (CI/CD) system that provides capabilities for automating safe rollouts. Teams are required to use appropriate safe roll-out strategies.

Common anti-patterns:

- You deploy an unsuccessful change to all of production all at once. As a result, all customers are impacted simultaneously.
- A defect introduced in a simultaneous deployment to all systems requires an emergency release. Correcting it for all customers takes several days.
- Managing production release requires planning and participation of several teams. This puts constraints on your ability to frequently update features for your customers.
- You perform a mutable deployment by modifying your existing systems. After discovering that the change was unsuccessful, you are forced to modify the systems again to restore the old version, extending your time to recovery.

Benefits of establishing this best practice: Automated deployments balance speed of roll-outs against delivering beneficial changes consistently to customers. Limiting impact prevents costly deployment failures and maximizes teams ability to efficiently respond to failures.

Level of risk exposed if this best practice is not established: Medium

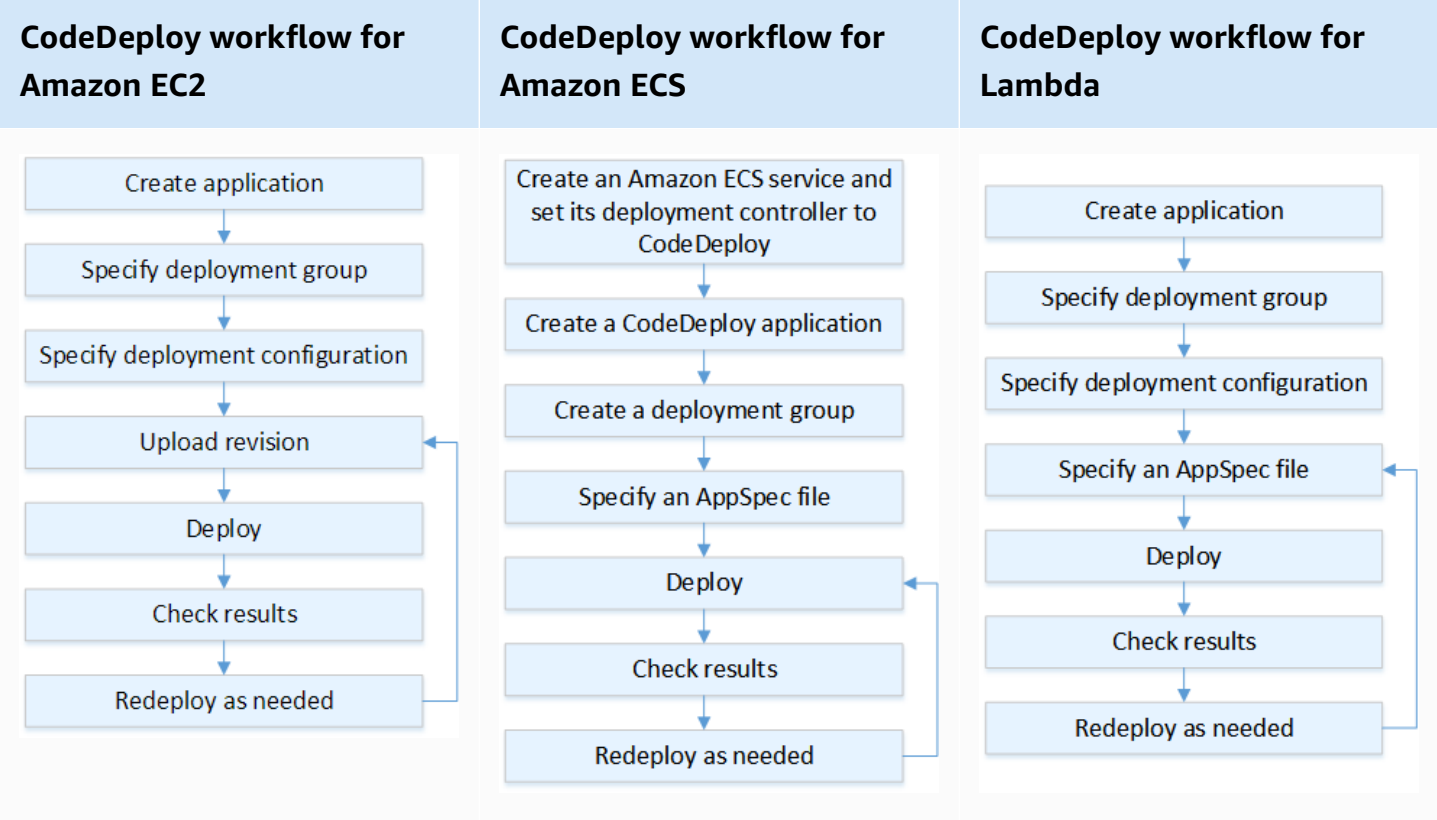
Implementation guidance

Continuous-delivery failures can lead to reduced service availability and bad customer experiences. To maximize the rate of successful deployments, implement safety controls in the end-to-end release process to minimize deployment errors, with a goal of achieving zero deployment failures.

Customer example

AnyCompany Retail is on a mission to achieve minimal to zero downtime deployments, meaning that there's no perceivable impact to its users during deployments. To accomplish this, the company has established deployment patterns (see the following workflow diagram), such as

rolling and blue/green deployments. All teams adopt one or more of these patterns in their CI/CD pipeline.



Implementation steps

1. Use an approval workflow to initiate the sequence of production roll-out steps upon promotion to production .
2. Use an automated deployment system such as AWS CodeDeploy. AWS CodeDeploy deployment options include in-place deployments for EC2/On-Premises and blue/green deployments for EC2/On-Premises, AWS Lambda, and Amazon ECS (see the preceding workflow diagram).
   a. Where applicable, integrate AWS CodeDeploy with other AWS services or integrate AWS CodeDeploy with partner product and services.
3. Use blue/green deployments for databases such as Amazon Aurora and Amazon RDS.
4. Monitor deployments using Amazon CloudWatch, AWS CloudTrail, and Amazon Simple Notification Service (Amazon SNS) event notifications.
5. Perform post-deployment automated testing including functional, security, regression, integration, and any load tests.

6. Troubleshoot deployment issues.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS05-BP02 Test and validate changes
- OPS05-BP09 Make frequent, small, reversible changes
- OPS05-BP10 Fully automate integration and deployment

Related documents:

- AWS Builders Library | Automating safe, hands-off deployments | Production deployments
- AWS Builders Library | My CI/CD pipeline is my release captain | Safe, automatic production releases
- AWS Whitepaper | Practicing Continuous Integration and Continuous Delivery on AWS | Deployment methods
- AWS CodeDeploy User Guide
- Working with deployment configurations in AWS CodeDeploy
- Set up an API Gateway canary release deployment
- Amazon ECS Deployment Types
- Fully Managed Blue/Green Deployments in Amazon Aurora and Amazon RDS
- Blue/Green deployments with AWS Elastic Beanstalk

Related videos:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

Related examples:

- Try a Sample Blue/Green Deployment in AWS CodeDeploy
- Workshop | Building CI/CD pipelines for Lambda canary deployments using AWS CDK

- Workshop | Building your first DevOps Blue/Green pipeline with Amazon ECS
- Workshop | Building your first DevOps Blue/Green pipeline with Amazon EKS
- Workshop | EKS GitOps with ArgoCD
- Workshop | CI/CD on AWS Workshop
- Implementing cross-account CI/CD with AWS SAM for container-based Lambda functions

OPS06-BP04 Automate testing and rollback

To increase the speed, reliability, and confidence of your deployment process, have a strategy for automated testing and rollback capabilities in pre-production and production environments. Automate testing when deploying to production to simulate human and system interactions that verify the changes being deployed. Automate rollback to revert back to a previous known good state quickly. The rollback should be initiated automatically on pre-defined conditions such as when the desired outcome of your change is not achieved or when the automated test fails. Automating these two activities improves your success rate for your deployments, minimizes recovery time, and reduces the potential impact to the business.

Desired outcome: Your automated tests and rollback strategies are integrated into your continuous integration, continuous delivery (CI/CD) pipeline. Your monitoring is able to validate against your success criteria and initiate automatic rollback upon failure. This minimizes any impact to end users and customers. For example, when all testing outcomes have been satisfied, you promote your code into the production environment where automated regression testing is initiated, leveraging the same test cases. If regression test results do not match expectations, then automated rollback is initiated in the pipeline workflow.

Common anti-patterns:

- Your systems are not architected in a way that allows them to be updated with smaller releases. As a result, you have difficulty in reversing those bulk changes during a failed deployment.
- Your deployment process consists of a series of manual steps. After you deploy changes to your workload, you start post-deployment testing. After testing, you realize that your workload is inoperable and customers are disconnected. You then begin rolling back to the previous version. All of these manual steps delay overall system recovery and cause a prolonged impact to your customers.
- You spent time developing automated test cases for functionality that is not frequently used in your application, minimizing the return on investment in your automated testing capability.

- Your release is comprised of application, infrastructure, patches and configuration updates that are independent from one another. However, you have a single CI/CD pipeline that delivers all changes at once. A failure in one component forces you to revert all changes, making your rollback complex and inefficient.
- Your team completes the coding work in sprint one and begins sprint two work, but your plan did not include testing until sprint three. As a result, automated tests revealed defects from sprint one that had to be resolved before testing of sprint two deliverables could be started and the entire release is delayed, devaluing your automated testing.
- Your automated regression test cases for the production release are complete, but you are not monitoring workload health. Since you have no visibility into whether or not the service has restarted, you are not sure if rollback is needed or if it has already occurred.

Benefits of establishing this best practice: Automated testing increases the transparency of your testing process and your ability to cover more features in a shorter time period. By testing and validating changes in production, you are able to identify issues immediately. Improvement in consistency with automated testing tools allows for better detection of defects. By automatically rolling back to the previous version, the impact on your customers is minimized. Automated rollback ultimately inspires more confidence in your deployment capabilities by reducing business impact. Overall, these capabilities reduce time-to-delivery while ensuring quality.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Automate testing of deployed environments to confirm desired outcomes more quickly. Automate rollback to a previous known good state when pre-defined outcomes are not achieved to minimize recovery time and reduce errors caused by manual processes. Integrate testing tools with your pipeline workflow to consistently test and minimize manual inputs. Prioritize automating test cases, such as those that mitigate the greatest risks and need to be tested frequently with every change. Additionally, automate rollback based on specific conditions that are pre-defined in your test plan.

Implementation steps

1. Establish a testing lifecycle for your development lifecycle that defines each stage of the testing process from requirements planning to test case development, tool configuration, automated testing, and test case closure.
   a. Create a workload-specific testing approach from your overall test strategy.

   b. Consider a continuous testing strategy where appropriate throughout the development lifecycle.

2. Select automated tools for testing and rollback based on your business requirements and pipeline investments.
3. Decide which test cases you wish to automate and which should be performed manually. These can be defined based on business value priority of the feature being tested. Align all team members to this plan and verify accountability for performing manual tests.
   a. Apply automated testing capabilities to specific test cases that make sense for automation, such as repeatable or frequently run cases, those that require repetitive tasks, or those that are required across multiple configurations.
   b. Define test automation scripts as well as the success criteria in the automation tool so continued workflow automation can be initiated when specific cases fail.
   c. Define specific failure criteria for automated rollback.
4. Prioritize test automation to drive consistent results with thorough test case development where complexity and human interaction have a higher risk of failure.
5. Integrate your automated testing and rollback tools into your CI/CD pipeline.
   a. Develop clear success criteria for your changes.
   b. Monitor and observe to detect these criteria and automatically reverse changes when specific rollback criteria are met.
6. Perform different types of automated production testing, such as:
   a. A/B testing to show results in comparison to the current version between two user testing groups.
   b. Canary testing that allows you to roll out your change to a subset of users before releasing it to all.
   c. Feature-flag testing which allows a single feature of the new version at a time to be flagged on and off from outside the application so that each new feature can be validated one at a time.
   d. Regression testing to verify new functionality with existing interrelated components.
7. Monitor the operational aspects of the application, transactions, and interactions with other applications and components. Develop reports to show success of changes by workload so that you can identify what parts of the automation and workflow can be further optimized.
   a. Develop test result reports that help you make quick decisions on whether or not rollback procedures should be invoked.

   b. Implement a strategy that allows for automated rollback based upon pre-defined failure conditions that result from one or more of your test methods.
8. Develop your automated test cases to allow for reusability across future repeatable changes.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS06-BP01 Plan for unsuccessful changes
- OPS06-BP02 Test deployments

Related documents:

- AWS Builders Library | Ensuring rollback safety during deployments
- Redeploy and rollback a deployment with AWS CodeDeploy
- 8 best practices when automating your deployments with AWS CloudFormation

Related examples:

- Serverless UI testing using Selenium, AWS Lambda, AWS Fargate, and AWS Developer Tools

Related videos:

- re:Invent 2020 | Hands-off: Automating continuous delivery pipelines at Amazon
- re:Invent 2019 | Amazon's Approach to high-availability deployment

OPS 7. How do you know that you are ready to support a workload?

Evaluate the operational readiness of your workload, processes and procedures, and personnel to understand the operational risks related to your workload.

Best practices

- OPS07-BP01 Ensure personnel capability
- OPS07-BP02 Ensure a consistent review of operational readiness

- OPS07-BP03 Use runbooks to perform procedures
- OPS07-BP04 Use playbooks to investigate issues
- OPS07-BP05 Make informed decisions to deploy systems and changes
- OPS07-BP06 Create support plans for production workloads

OPS07-BP01 Ensure personnel capability

Have a mechanism to validate that you have the appropriate number of trained personnel to support the workload. They must be trained on the platform and services that make up your workload. Provide them with the knowledge necessary to operate the workload. You must have enough trained personnel to support the normal operation of the workload and troubleshoot any incidents that occur. Have enough personnel so that you can rotate during on-call and vacations to avoid burnout.

Desired outcome:

- There are enough trained personnel to support the workload at times when the workload is available.
- You provide training for your personnel on the software and services that make up your workload.

Common anti-patterns:

- Deploying a workload without team members trained to operate the platform and services in use.
- Not having enough personnel to support on-call rotations or personnel taking time off.

Benefits of establishing this best practice:

- Having skilled team members helps effective support of your workload.
- With enough team members, you can support the workload and on-call rotations while decreasing the risk of burnout.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Validate that there are sufficient trained personnel to support the workload. Verify that you have enough team members to cover normal operational activities, including on-call rotations.

Customer example

AnyCompany Retail makes sure that teams supporting the workload are properly staffed and trained. They have enough engineers to support an on-call rotation. Personnel get training on the software and platform that the workload is built on and are encouraged to earn certifications. There are enough personnel so that people can take time off while still supporting the workload and the on-call rotation.

Implementation steps

1. Assign an adequate number of personnel to operate and support your workload, including on-call duties, security issues, and lifecycle events, such as end of support and certificate rotation tasks.
2. Train your personnel on the software and platforms that compose your workload.
   a. AWS Training and Certification has a library of courses about AWS. They provide free and paid courses, online and in-person.
   b. AWS hosts events and webinars where you learn from AWS experts.
3. Perform the following on a regular basis:
   - Evaluate team size and skills as operating conditions and the workload change.
   - Adjust team size and skills to match operational requirements.
   - Verify ability and capacity to address planned lifecycle events, unplanned security, and operational notifications through AWS Health.

Level of effort for the implementation plan: High. Hiring and training a team to support a workload can take significant effort but has substantial long-term benefits.

Resources

Related best practices:

- OPS11-BP04 Perform knowledge management - Team members must have the information necessary to operate and support the workload. Knowledge management is the key to providing that.

Related documents:

- AWS Events and Webinars
- AWS Training and Certification

OPS07-BP02 Ensure a consistent review of operational readiness

Use Operational Readiness Reviews (ORRs) to validate that you can operate your workload. ORR is a mechanism developed at Amazon to validate that teams can safely operate their workloads. An ORR is a review and inspection process using a checklist of requirements. An ORR is a self-service experience that teams use to certify their workloads. ORRs include best practices from lessons learned from our years of building software.

An ORR checklist is composed of architectural recommendations, operational process, event management, and release quality. Our Correction of Error (CoE) process is a major driver of these items. Your own post-incident analysis should drive the evolution of your own ORR. An ORR is not only about following best practices but preventing the recurrence of events that you've seen before. Lastly, security, governance, and compliance requirements can also be included in an ORR.

Run ORRs before a workload launches to general availability and then throughout the software development lifecycle. Running the ORR before launch increases your ability to operate the workload safely. Periodically re-run your ORR on the workload to catch any drift from best practices. You can have ORR checklists for new services launches and ORRs for periodic reviews. This helps keep you up to date on new best practices that arise and incorporate lessons learned from post-incident analysis. As your use of the cloud matures, you can build ORR requirements into your architecture as defaults.

Desired outcome: You have an ORR checklist with best practices for your organization. ORRs are conducted before workloads launch. ORRs are run periodically over the course of the workload lifecycle.

Common anti-patterns:

- You launch a workload without knowing if you can operate it.
- Governance and security requirements are not included in certifying a workload for launch.
- Workloads are not re-evaluated periodically.
- Workloads launch without required procedures in place.

- You see repetition of the same root cause failures in multiple workloads.

Benefits of establishing this best practice:

- Your workloads include architecture, process, and management best practices.
- Lessons learned are incorporated into your ORR process.
- Required procedures are in place when workloads launch.
- ORRs are run throughout the software lifecycle of your workloads.

Level of risk if this best practice is not established: High

Implementation guidance

An ORR is two things: a process and a checklist. Your ORR process should be adopted by your organization and supported by an executive sponsor. At a minimum, ORRs must be conducted before a workload launches to general availability. Run the ORR throughout the software development lifecycle to keep it up to date with best practices or new requirements. The ORR checklist should include configuration items, security and governance requirements, and best practices from your organization. Over time, you can use services, such as AWS Config, AWS Security Hub CSPM, and AWS Control Tower Guardrails, to build best practices from the ORR into guardrails for automatic detection of best practices.

Customer example

After several production incidents, AnyCompany Retail decided to implement an ORR process. They built a checklist composed of best practices, governance and compliance requirements, and lessons learned from outages. New workloads conduct ORRs before they launch. Every workload conducts a yearly ORR with a subset of best practices to incorporate new best practices and requirements that are added to the ORR checklist. Over time, AnyCompany Retail used AWS Config to detect some best practices, speeding up the ORR process.

Implementation steps

To learn more about ORRs, read the Operational Readiness Reviews (ORR) whitepaper. It provides detailed information on the history of the ORR process, how to build your own ORR practice, and how to develop your ORR checklist. The following steps are an abbreviated version of that document. For an in-depth understanding of what ORRs are and how to build your own, we recommend reading that whitepaper.

1. Gather the key stakeholders together, including representatives from security, operations, and development.
2. Have each stakeholder provide at least one requirement. For the first iteration, try to limit the number of items to thirty or less.
   - Appendix B: Example ORR questions from the Operational Readiness Reviews (ORR) whitepaper contains sample questions that you can use to get started.
3. Collect your requirements into a spreadsheet.
   - You can use custom lenses in the AWS Well-Architected Tool to develop your ORR and share them across your accounts and AWS Organization.
4. Identify one workload to conduct the ORR on. A pre-launch workload or an internal workload is ideal.
5. Run through the ORR checklist and take note of any discoveries made. Discoveries might be acceptable if a mitigation is in place. For any discovery that lacks a mitigation, add those to your backlog of items and implement them before launch.
6. Continue to add best practices and requirements to your ORR checklist over time.

Support customers with Enterprise Support can request the Operational Readiness Review Workshop from their Technical Account Manager. The workshop is an interactive *working backwards* session to develop your own ORR checklist.

Level of effort for the implementation plan: High. Adopting an ORR practice in your organization requires executive sponsorship and stakeholder buy-in. Build and update the checklist with inputs from across your organization.

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements – Governance requirements are a natural fit for an ORR checklist.
- OPS01-BP04 Evaluate compliance requirements – Compliance requirements are sometimes included in an ORR checklist. Other times they are a separate process.
- OPS03-BP07 Resource teams appropriately – Team capability is a good candidate for an ORR requirement.
- OPS06-BP01 Plan for unsuccessful changes – A rollback or rollforward plan must be established before you launch your workload.

- OPS07-BP01 Ensure personnel capability – To support a workload you must have the required personnel.
- SEC01-BP03 Identify and validate control objectives – Security control objectives make excellent ORR requirements.
- REL13-BP01 Define recovery objectives for downtime and data loss – Disaster recovery plans are a good ORR requirement.
- COST02-BP01 Develop policies based on your organization requirements – Cost management policies are good to include in your ORR checklist.

Related documents:

- AWS Control Tower - Guardrails in AWS Control Tower
- AWS Well-Architected Tool - Custom Lenses
- Operational Readiness Review Template by Adrian Hornsby
- Operational Readiness Reviews (ORR) Whitepaper

Related videos:

- AWS Supports You | Building an Effective Operational Readiness Review (ORR)

Related examples:

- Sample Operational Readiness Review (ORR) Lens

Related services:

- AWS Config
- AWS Control Tower
- AWS Security Hub CSPM
- AWS Well-Architected Tool

OPS07-BP03 Use runbooks to perform procedures

A *runbook* is a documented process to achieve a specific outcome. Runbooks consist of a series of steps that someone follows to get something done. Runbooks have been used in operations going

back to the early days of aviation. In cloud operations, we use runbooks to reduce risk and achieve desired outcomes. At its simplest, a runbook is a checklist to complete a task.

Runbooks are an essential part of operating your workload. From onboarding a new team member to deploying a major release, runbooks are the codified processes that provide consistent outcomes no matter who uses them. Runbooks should be published in a central location and updated as the process evolves, as updating runbooks is a key component of a change management process. They should also include guidance on error handling, tools, permissions, exceptions, and escalations in case a problem occurs.

As your organization matures, begin automating runbooks. Start with runbooks that are short and frequently used. Use scripting languages to automate steps or make steps easier to perform. As you automate the first few runbooks, you'll dedicate time to automating more complex runbooks. Over time, most of your runbooks should be automated in some way.

Desired outcome: Your team has a collection of step-by-step guides for performing workload tasks. The runbooks contain the desired outcome, necessary tools and permissions, and instructions for error handling. They are stored in a central location (version control system) and updated frequently. For example, your runbooks provide capabilities for your teams to monitor, communicate, and respond to AWS Health events for critical accounts during application alarms, operational issues, and planned lifecycle events.

Common anti-patterns:

- Relying on memory to complete each step of a process.
- Manually deploying changes without a checklist.
- Different team members performing the same process but with different steps or outcomes.
- Letting runbooks drift out of sync with system changes and automation.

Benefits of establishing this best practice:

- Reducing error rates for manual tasks.
- Operations are performed in a consistent manner.
- New team members can start performing tasks sooner.
- Runbooks can be automated to reduce toil.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Runbooks can take several forms depending on the maturity level of your organization. At a minimum, they should consist of a step-by-step text document. The desired outcome should be clearly indicated. Clearly document necessary special permissions or tools. Provide detailed guidance on error handling and escalations in case something goes wrong. List the runbook owner and publish it in a central location. Once your runbook is documented, validate it by having someone else on your team run it. As procedures evolve, update your runbooks in accordance with your change management process.

Your text runbooks should be automated as your organization matures. Using services like AWS Systems Manager automations, you can transform flat text into automations that can be run against your workload. These automations can be run in response to events, reducing the operational burden to maintain your workload. AWS Systems Manager Automation also provides a low-code visual design experience to create automation runbooks more easily.

Customer example

AnyCompany Retail must perform database schema updates during software deployments. The Cloud Operations Team worked with the Database Administration Team to build a runbook for manually deploying these changes. The runbook listed each step in the process in checklist form. It included a section on error handling in case something went wrong. They published the runbook on their internal wiki along with their other runbooks. The Cloud Operations Team plans to automate the runbook in a future sprint.

Implementation steps

If you don't have an existing document repository, a version control repository is a great place to start building your runbook library. You can build your runbooks using Markdown. We have provided an example runbook template that you can use to start building runbooks.

```
# Runbook Title
## Runbook Info
| Runbook ID | Description | Tools Used | Special Permissions | Runbook Author | Last
 Updated | Escalation POC |
|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this runbook for? What is the desired outcome? | Tools | Permissions
 | Your Name | 2022-09-21 | Escalation Name |
## Steps
1. Step one
```

```
2. Step two
```

1. If you don't have an existing documentation repository or wiki, create a new version control repository in your version control system.
2. Identify a process that does not have a runbook. An ideal process is one that is conducted semiregularly, short in number of steps, and has low impact failures.
3. In your document repository, create a new draft Markdown document using the template. Fill in Runbook Title and the required fields under Runbook Info.
4. Starting with the first step, fill in the Steps portion of the runbook.
5. Give the runbook to a team member. Have them use the runbook to validate the steps. If something is missing or needs clarity, update the runbook.
6. Publish the runbook to your internal documentation store. Once published, tell your team and other stakeholders.
7. Over time, you'll build a library of runbooks. As that library grows, start working to automate runbooks.

Level of effort for the implementation plan: Low. The minimum standard for a runbook is a step-by-step text guide. Automating runbooks can increase the implementation effort.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners
- OPS07-BP04 Use playbooks to investigate issues
- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP02 Have a process per alert
- OPS11-BP04 Perform knowledge management

Related documents:

- Achieving Operational Excellence using automated playbook and runbook
- AWS Systems Manager: Working with runbooks
- Migration playbook for AWS large migrations - Task 4: Improving your migration runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

Related videos:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response
- How to automate IT Operations on AWS | Amazon Web Services
- Integrate Scripts into AWS Systems Manager

Related examples:

- Well-Architected Labs: Automating operations with Playbooks and Runbooks
- AWS Blog Post: Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- AWS Systems Manager: Automation walkthroughs
- AWS Systems Manager: Restore a root volume from the latest snapshot runbook
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Gitlab - Runbooks
- Rubix - A Python library for building runbooks in Jupyter Notebooks
- Using Document Builder to create a custom runbook

Related services:

- AWS Systems Manager Automation

OPS07-BP04 Use playbooks to investigate issues

Playbooks are step-by-step guides used to investigate an incident. When incidents happen, playbooks are used to investigate, scope impact, and identify a root cause. Playbooks are used for a variety of scenarios, from failed deployments to security incidents. In many cases, playbooks identify the root cause that a runbook is used to mitigate. Playbooks are an essential component of your organization's incident response plans.

A good playbook has several key features. It guides the user, step by step, through the process of discovery. Thinking outside-in, what steps should someone follow to diagnose an incident? Clearly define in the playbook if special tools or elevated permissions are needed in the playbook. Having a communication plan to update stakeholders on the status of the investigation is a key component. In situations where a root cause can't be identified, the playbook should have an escalation plan. If

the root cause is identified, the playbook should point to a runbook that describes how to resolve it. Playbooks should be stored centrally and regularly maintained. If playbooks are used for specific alerts, provide your team with pointers to the playbook within the alert.

As your organization matures, automate your playbooks. Start with playbooks that cover low-risk incidents. Use scripting to automate the discovery steps. Make sure that you have companion runbooks to mitigate common root causes.

Desired outcome: Your organization has playbooks for common incidents. The playbooks are stored in a central location and available to your team members. Playbooks are updated frequently. For any known root causes, companion runbooks are built.

Common anti-patterns:

- There is no standard way to investigate an incident.
- Team members rely on muscle memory or institutional knowledge to troubleshoot a failed deployment.
- New team members learn how to investigate issues through trial and error.
- Best practices for investigating issues are not shared across teams.

Benefits of establishing this best practice:

- Playbooks boost your efforts to mitigate incidents.
- Different team members can use the same playbook to identify a root cause in a consistent manner.
- Known root causes can have runbooks developed for them, speeding up recovery time.
- Playbooks help team members to start contributing sooner.
- Teams can scale their processes with repeatable playbooks.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

How you build and use playbooks depends on the maturity of your organization. If you are new to the cloud, build playbooks in text form in a central document repository. As your organization matures, playbooks can become semi-automated with scripting languages like Python. These

scripts can be run inside a Jupyter notebook to speed up discovery. Advanced organizations have fully automated playbooks for common issues that are auto-remediated with runbooks.

Start building your playbooks by listing common incidents that happen to your workload. Choose playbooks for incidents that are low risk and where the root cause has been narrowed down to a few issues to start. After you have playbooks for simpler scenarios, move on to the higher risk scenarios or scenarios where the root cause is not well known.

Your text playbooks should be automated as your organization matures. Using services like AWS Systems Manager Automations, flat text can be transformed into automations. These automations can be run against your workload to speed up investigations. These automations can be activated in response to events, reducing the mean time to discover and resolve incidents.

Customers can use AWS Systems Manager Incident Manager to respond to incidents. This service provides a single interface to triage incidents, inform stakeholders during discovery and mitigation, and collaborate throughout the incident. It uses AWS Systems Manager Automations to speed up detection and recovery.

Customer example

A production incident impacted AnyCompany Retail. The on-call engineer used a playbook to investigate the issue. As they progressed through the steps, they kept the key stakeholders, identified in the playbook, up to date. The engineer identified the root cause as a race condition in a backend service. Using a runbook, the engineer relaunched the service, bringing AnyCompany Retail back online.

Implementation steps

If you don't have an existing document repository, we suggest creating a version control repository for your playbook library. You can build your playbooks using Markdown, which is compatible with most playbook automation systems. If you are starting from scratch, use the following example playbook template.

```
# Playbook Title
## Playbook Info
| Playbook ID | Description | Tools Used | Special Permissions | Playbook Author | Last
 Updated | Escalation POC | Stakeholders | Communication Plan |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| RUN001 | What is this playbook for? What incident is it used for? | Tools |
 Permissions | Your Name | 2022-09-21 | Escalation Name | Stakeholder Name | How will
 updates be communicated during the investigation? |
```

```
## Steps
1. Step one
2. Step two
```

1. If you don't have an existing document repository or wiki, create a new version control repository for your playbooks in your version control system.
2. Identify a common issue that requires investigation. This should be a scenario where the root cause is limited to a few issues and resolution is low risk.
3. Using the Markdown template, fill in the Playbook Name section and the fields under Playbook Info.
4. Fill in the troubleshooting steps. Be as clear as possible on what actions to perform or what areas you should investigate.
5. Give a team member the playbook and have them go through it to validate it. If there's anything missing or something isn't clear, update the playbook.
6. Publish your playbook in your document repository and inform your team and any stakeholders.
7. This playbook library will grow as you add more playbooks. Once you have several playbooks, start automating them using tools like AWS Systems Manager Automations to keep automation and playbooks in sync.

Level of effort for the implementation plan: Low. Your playbooks should be text documents stored in a central location. More mature organizations will move towards automating playbooks.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners
- OPS07-BP03 Use runbooks to perform procedures
- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP02 Have a process per alert
- OPS11-BP04 Perform knowledge management

Related documents:

- Achieving Operational Excellence using automated playbook and runbook

- AWS Systems Manager: Working with runbooks
- Use AWS Systems Manager Automation runbooks to resolve operational tasks

Related videos:

- AWS re:Invent 2019: DIY guide to runbooks, incident reports, and incident response (SEC318-R1)
- AWS Systems Manager Incident Manager - AWS Virtual Workshops
- Integrate Scripts into AWS Systems Manager

Related examples:

- AWS Customer Playbook Framework
- AWS Systems Manager: Automation walkthroughs
- Building an AWS incident response runbook using Jupyter notebooks and CloudTrail Lake
- Rubix – A Python library for building runbooks in Jupyter Notebooks
- Using Document Builder to create a custom runbook

Related services:

- AWS Systems Manager Automation
- AWS Systems Manager Incident Manager

OPS07-BP05 Make informed decisions to deploy systems and changes

Have processes in place for successful and unsuccessful changes to your workload. A pre-mortem is an exercise where a team simulates a failure to develop mitigation strategies. Use pre-mortems to anticipate failure and create procedures where appropriate. Evaluate the benefits and risks of deploying changes to your workload. Verify that all changes comply with governance.

Desired outcome:

- You make informed decisions when deploying changes to your workload.
- Changes comply with governance.

Common anti-patterns:

- Deploying a change to our workload without a process to handle a failed deployment.
- Making changes to your production environment that are out of compliance with governance requirements.
- Deploying a new version of your workload without establishing a baseline for resource utilization.

Benefits of establishing this best practice:

- You are prepared for unsuccessful changes to your workload.
- Changes to your workload are compliant with governance policies.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use pre-mortems to develop processes for unsuccessful changes. Document your processes for unsuccessful changes. Ensure that all changes comply with governance. Evaluate the benefits and risks to deploying changes to your workload.

Customer example

AnyCompany Retail regularly conducts pre-mortems to validate their processes for unsuccessful changes. They document their processes in a shared Wiki and update it frequently. All changes comply with governance requirements.

Implementation steps

1. Make informed decisions when deploying changes to your workload. Establish and review criteria for a successful deployment. Develop scenarios or criteria that would initiate a rollback of a change. Weigh the benefits of deploying changes against the risks of an unsuccessful change.
2. Verify that all changes comply with governance policies.
3. Use pre-mortems to plan for unsuccessful changes and document mitigation strategies. Run a table-top exercise to model an unsuccessful change and validate roll-back procedures.

Level of effort for the implementation plan: Moderate. Implementing a practice of pre-mortems requires coordination and effort from stakeholders across your organization

Resources

Related best practices:

- OPS01-BP03 Evaluate governance requirements - Governance requirements are a key factor in determining whether to deploy a change.
- OPS06-BP01 Plan for unsuccessful changes - Establish plans to mitigate a failed deployment and use pre-mortems to validate them.
- OPS06-BP02 Test deployments - Every software change should be properly tested before deployment in order to reduce defects in production.
- OPS07-BP01 Ensure personnel capability - Having enough trained personnel to support the workload is essential to making an informed decision to deploy a system change.

Related documents:

- Amazon Web Services: Risk and Compliance
- AWS Shared Responsibility Model
- Governance in the AWS Cloud: The Right Balance Between Agility and Safety

OPS07-BP06 Create support plans for production workloads

Enable support for any software and services that your production workload relies on. Select an appropriate support level to meet your production service-level needs. Support plans for these dependencies are necessary in case there is a service disruption or software issue. Document support plans and how to request support for all service and software vendors. Implement mechanisms that verify that support points of contacts are kept up to date.

Desired outcome:

- Implement support plans for software and services that production workloads rely on.
- Choose an appropriate support plan based on service-level needs.
- Document the support plans, support levels, and how to request support.

Common anti-patterns:

- You have no support plan for a critical software vendor. Your workload is impacted by them and you can do nothing to expedite a fix or get timely updates from the vendor.

- A developer that was the primary point of contact for a software vendor left the company. You are not able to reach the vendor support directly. You must spend time researching and navigating generic contact systems, increasing the time required to respond when needed.
- A production outage occurs with a software vendor. There is no documentation on how to file a support case.

Benefits of establishing this best practice:

- With the appropriate support level, you are able to get a response in the time frame necessary to meet service-level needs.
- As a supported customer you can escalate if there are production issues.
- Software and services vendors can assist in troubleshooting during an incident.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Enable support plans for any software and services vendors that your production workload relies on. Set up appropriate support plans to meet service-level needs. For AWS customers, this means activating AWS Business Support or greater on any accounts where you have production workloads. Meet with support vendors on a regular cadence to get updates about support offerings, processes, and contacts. Document how to request support from software and services vendors, including how to escalate if there is an outage. Implement mechanisms to keep support contacts up to date.

Customer example

At AnyCompany Retail, all commercial software and services dependencies have support plans. For example, they have AWS Enterprise Support activated on all accounts with production workloads. Any developer can raise a support case when there is an issue. There is a wiki page with information on how to request support, whom to notify, and best practices for expediting a case.

Implementation steps

1. Work with stakeholders in your organization to identify software and services vendors that your workload relies on. Document these dependencies.
2. Determine service-level needs for your workload. Select a support plan that aligns with them.
3. For commercial software and services, establish a support plan with the vendors.

   a. Subscribing to AWS Business Support or greater for all production accounts provides faster response time from AWS Support and strongly recommended. If you don't have premium support, you must have an action plan to handle issues, which require help from AWS Support. AWS Support provides a mix of tools and technology, people, and programs designed to proactively help you optimize performance, lower costs, and innovate faster. In addition, AWS Business Support provides additional benefits, including API access to AWS Trusted Advisor and AWS Health for programmatic integration with your systems, alongside other access methods like the AWS Management Console and Amazon EventBridge channels.

4. Document the support plan in your knowledge management tool. Include how to request support, who to notify if a support case is filed, and how to escalate during an incident. A wiki is a good mechanism to allow anyone to make necessary updates to documentation when they become aware of changes to support processes or contacts.

Level of effort for the implementation plan: Low. Most software and services vendors offer opt-in support plans. Documenting and sharing support best practices on your knowledge management system verifies that your team knows what to do when there is a production issue.

Resources

Related best practices:

- OPS02-BP02 Processes and procedures have identified owners

Related documents:

- AWS Support Plans

Related services:

- AWS Business Support
- AWS Enterprise Support

Operate

Questions

- OPS 8. How do you utilize workload observability in your organization?

- OPS 9. How do you understand the health of your operations?
- OPS 10. How do you manage workload and operations events?

OPS 8. How do you utilize workload observability in your organization?

Ensure optimal workload health by leveraging observability. Utilize relevant metrics, logs, and traces to gain a comprehensive view of your workload's performance and address issues efficiently.

Best practices

- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs
- OPS08-BP03 Analyze workload traces
- OPS08-BP04 Create actionable alerts
- OPS08-BP05 Create dashboards

OPS08-BP01 Analyze workload metrics

After implementing application telemetry, regularly analyze the collected metrics. While latency, requests, errors, and capacity (or quotas) provide insights into system performance, it's vital to prioritize the review of business outcome metrics. This ensures you're making data-driven decisions aligned with your business objectives.

Desired outcome: Accurate insights into workload performance that drive data-informed decisions, ensuring alignment with business objectives.

Common anti-patterns:

- Analyzing metrics in isolation without considering their impact on business outcomes.
- Over-reliance on technical metrics while sidelining business metrics.
- Infrequent review of metrics, missing out on real-time decision-making opportunities.

Benefits of establishing this best practice:

- Enhanced understanding of the correlation between technical performance and business outcomes.

- Improved decision-making process informed by real-time data.
- Proactive identification and mitigation of issues before they affect business outcomes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Leverage tools like Amazon CloudWatch to perform metric analysis. AWS services such as CloudWatch anomaly detection and Amazon DevOps Guru can be used to detect anomalies, especially when static thresholds are unknown or when patterns of behavior are more suited for anomaly detection.

Implementation steps

1. **Analyze and review:** Regularly review and interpret your workload metrics.
   a. Prioritize business outcome metrics over purely technical metrics.
   b. Understand the significance of spikes, drops, or patterns in your data.
2. **Utilize Amazon CloudWatch:** Use Amazon CloudWatch for a centralized view and deep-dive analysis.
   a. Configure CloudWatch dashboards to visualize your metrics and compare them over time.
   b. Use percentiles in CloudWatch to get a clear view of metric distribution, which can help in defining SLAs and understanding outliers.
   c. Set up CloudWatch anomaly detection to identify unusual patterns without relying on static thresholds.
   d. Implement CloudWatch cross-account observability to monitor and troubleshoot applications that span multiple accounts within a Region.
   e. Use CloudWatch Metric Insights to query and analyze metric data across accounts and Regions, identifying trends and anomalies.
   f. Apply CloudWatch Metric Math to transform, aggregate, or perform calculations on your metrics for deeper insights.
3. **Employ Amazon DevOps Guru:** Incorporate Amazon DevOps Guru for its machine learning-enhanced anomaly detection to identify early signs of operational issues for your serverless applications and remediate them before they impact your customers.
4. **Optimize based on insights:** Make informed decisions based on your metric analysis to adjust and improve your workloads.

Level of effort for the Implementation Plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry

Related documents:

- The Wheel Blog - Emphasizing the importance of continually reviewing metrics
- Percentile are important
- Using AWS Cost Anomaly Detection
- CloudWatch cross-account observability
- Query your metrics with CloudWatch Metrics Insights

Related videos:

- Enable Cross-Account Observability in Amazon CloudWatch
- Introduction to Amazon DevOps Guru
- Continuously Analyze Metrics using AWS Cost Anomaly Detection

Related examples:

- One Observability Workshop
- Gaining operation insights with AIOps using Amazon DevOps Guru

OPS08-BP02 Analyze workload logs

Regularly analyzing workload logs is essential for gaining a deeper understanding of the operational aspects of your application. By efficiently sifting through, visualizing, and interpreting log data, you can continually optimize application performance and security.

Desired outcome: Rich insights into application behavior and operations derived from thorough log analysis, ensuring proactive issue detection and mitigation.

Common anti-patterns:

- Neglecting the analysis of logs until a critical issue arises.
- Not using the full suite of tools available for log analysis, missing out on critical insights.
- Solely relying on manual review of logs without leveraging automation and querying capabilities.

Benefits of establishing this best practice:

- Proactive identification of operational bottlenecks, security threats, and other potential issues.
- Efficient utilization of log data for continuous application optimization.
- Enhanced understanding of application behavior, aiding in debugging and troubleshooting.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Amazon CloudWatch Logs is a powerful tool for log analysis. Integrated features like CloudWatch Logs Insights and Contributor Insights make the process of deriving meaningful information from logs intuitive and efficient.

Implementation steps

1. **Set up CloudWatch Logs**: Configure applications and services to send logs to CloudWatch Logs.
2. **Use log anomaly detection:** Utilize Amazon CloudWatch Logs anomaly detection to automatically identify and alert on unusual log patterns. This tool helps you proactively manage anomalies in your logs and detect potential issues early.
3. **Set up CloudWatch Logs Insights**: Use CloudWatch Logs Insights to interactively search and analyze your log data.
    a. Craft queries to extract patterns, visualize log data, and derive actionable insights.
    b. Use CloudWatch Logs Insights pattern analysis to analyze and visualize frequent log patterns. This feature helps you understand common operational trends and potential outliers in your log data.
    c. Use CloudWatch Logs compare (diff) to perform differential analysis between different time periods or across different log groups. Use this capability to pinpoint changes and assess their impacts on your system's performance or behavior.

4. **Monitor logs in real-time with Live Tail:** Use Amazon CloudWatch Logs Live Tail to view log data in real-time. You can actively monitor your application's operational activities as they occur, which provides immediate visibility into system performance and potential issues.
5. **Leverage Contributor Insights**: Use CloudWatch Contributor Insights to identify top talkers in high cardinality dimensions like IP addresses or user-agents.
6. **Implement CloudWatch Logs metric filters**: Configure CloudWatch Logs metric filters to convert log data into actionable metrics. This allows you to set alarms or further analyze patterns.
7. **Implement CloudWatch cross-account observability:** Monitor and troubleshoot applications that span multiple accounts within a Region.
8. **Regular review and refinement**: Periodically review your log analysis strategies to capture all relevant information and continually optimize application performance.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS08-BP01 Analyze workload metrics

Related documents:

- Analyzing Log Data with CloudWatch Logs Insights
- Using CloudWatch Contributor Insights
- Creating and Managing CloudWatch Log Metric Filters

Related videos:

- Analyze Log Data with CloudWatch Logs Insights
- Use CloudWatch Contributor Insights to Analyze High-Cardinality Data

Related examples:

- CloudWatch Logs Sample Queries
- One Observability Workshop

OPS08-BP03 Analyze workload traces

Analyzing trace data is crucial for achieving a comprehensive view of an application's operational journey. By visualizing and understanding the interactions between various components, performance can be fine-tuned, bottlenecks identified, and user experiences enhanced.

Desired outcome: Achieve clear visibility into your application's distributed operations, enabling quicker issue resolution and an enhanced user experience.

Common anti-patterns:

- Overlooking trace data, relying solely on logs and metrics.
- Not correlating trace data with associated logs.
- Ignoring the metrics derived from traces, such as latency and fault rates.

Benefits of establishing this best practice:

- Improve troubleshooting and reduce mean time to resolution (MTTR).
- Gain insights into dependencies and their impact.
- Swift identification and rectification of performance issues.
- Leveraging trace-derived metrics for informed decision-making.
- Improved user experiences through optimized component interactions.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS X-Ray offers a comprehensive suite for trace data analysis, providing a holistic view of service interactions, monitoring user activities, and detecting performance issues. Features like ServiceLens, X-Ray Insights, X-Ray Analytics, and Amazon DevOps Guru enhance the depth of actionable insights derived from trace data.

Implementation steps

The following steps offer a structured approach to effectively implementing trace data analysis using AWS services:

1. **Integrate AWS X-Ray**: Ensure X-Ray is integrated with your applications to capture trace data.
2. **Analyze X-Ray metrics**: Delve into metrics derived from X-Ray traces, such as latency, request rates, fault rates, and response time distributions, using the service map to monitor application health.
3. **Use ServiceLens**: Leverage the ServiceLens map for enhanced observability of your services and applications. This allows for integrated viewing of traces, metrics, logs, alarms, and other health information.
4. **Enable X-Ray Insights**:
    a. Turn on X-Ray Insights for automated anomaly detection in traces.
    b. Examine insights to pinpoint patterns and ascertain root causes, such as increased fault rates or latencies.
    c. Consult the insights timeline for a chronological analysis of detected issues.
5. **Use X-Ray Analytics**: X-Ray Analytics allows you to thoroughly explore trace data, pinpoint patterns, and extract insights.
6. **Use groups in X-Ray**: Create groups in X-Ray to filter traces based on criteria such as high latency, allowing for more targeted analysis.
7. **Incorporate Amazon DevOps Guru**: Engage Amazon DevOps Guru to benefit from machine learning models pinpointing operational anomalies in traces.
8. **Use CloudWatch Synthetics**: Use CloudWatch Synthetics to create canaries for continually monitoring your endpoints and workflows. These canaries can integrate with X-Ray to provide trace data for in-depth analysis of the applications being tested.
9. **Use Real User Monitoring (RUM)**: With AWS X-Ray and CloudWatch RUM, you can analyze and debug the request path starting from end users of your application through downstream AWS managed services. This helps you identify latency trends and errors that impact your end users.
10. **Correlate with logs**: Correlate trace data with related logs within the X-Ray trace view for a granular perspective on application behavior. This allows you to view log events directly associated with traced transactions.
11. **Implement CloudWatch cross-account observability:** Monitor and troubleshoot applications that span multiple accounts within a Region.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs

Related documents:

- Using ServiceLens to Monitor Application Health
- Exploring Trace Data with X-Ray Analytics
- Detecting Anomalies in Traces with X-Ray Insights
- Continuous Monitoring with CloudWatch Synthetics

Related videos:

- Analyze and Debug Applications Using Amazon CloudWatch Synthetics & AWS X-Ray
- Use AWS X-Ray Insights

Related examples:

- One Observability Workshop
- Implementing X-Ray with AWS Lambda
- CloudWatch Synthetics Canary Templates

OPS08-BP04 Create actionable alerts

Promptly detecting and responding to deviations in your application's behavior is crucial. Especially vital is recognizing when outcomes based on key performance indicators (KPIs) are at risk or when unexpected anomalies arise. Basing alerts on KPIs ensures that the signals you receive are directly tied to business or operational impact. This approach to actionable alerts promotes proactive responses and helps maintain system performance and reliability.

Desired outcome: Receive timely, relevant, and actionable alerts for rapid identification and mitigation of potential issues, especially when KPI outcomes are at risk.

Common anti-patterns:

- Setting up too many non-critical alerts, leading to alert fatigue.
- Not prioritizing alerts based on KPIs, making it hard to understand the business impact of issues.
- Neglecting to address root causes, leading to repetitive alerts for the same issue.

Benefits of establishing this best practice:

- Reduced alert fatigue by focusing on actionable and relevant alerts.
- Improved system uptime and reliability through proactive issue detection and mitigation.
- Enhanced team collaboration and quicker issue resolution by integrating with popular alerting and communication tools.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To create an effective alerting mechanism, it's vital to use metrics, logs, and trace data that flag when outcomes based on KPIs are at risk or anomalies are detected.

Implementation steps

1. **Determine key performance indicators (KPIs)**: Identify your application's KPIs. Alerts should be tied to these KPIs to reflect the business impact accurately.
2. **Implement anomaly detection**:
   - **Use Amazon CloudWatch anomaly detection**: Set up Amazon CloudWatch anomaly detection to automatically detect unusual patterns, which helps you only generate alerts for genuine anomalies.
   - **Use AWS X-Ray Insights**:
     a. Set up X-Ray Insights to detect anomalies in trace data.
     b. Configure notifications for X-Ray Insights to be alerted on detected issues.
   - **Integrate with Amazon DevOps Guru**:
     a. Leverage Amazon DevOps Guru for its machine learning capabilities in detecting operational anomalies with existing data.
     b. Navigate to the notification settings in DevOps Guru to set up anomaly alerts.

3. **Implement actionable alerts**: Design alerts that provide adequate information for immediate action.
    1. Monitor AWS Health events with Amazon EventBridge rules, or integrate programatically with the AWS Health API to automate actions when you receive AWS Health events. These can be general actions, such as sending all planned lifecycle event messages to a chat interface, or specific actions, such as the initiation of a workflow in an IT service management tool.
4. **Reduce alert fatigue**: Minimize non-critical alerts. When teams are overwhelmed with numerous insignificant alerts, they can lose oversight of critical issues, which diminishes the overall effectiveness of the alert mechanism.
5. **Set up composite alarms**: Use Amazon CloudWatch composite alarms to consolidate multiple alarms.
6. **Integrate with alert tools**: Incorporate tools like Ops Genie and PagerDuty.
7. **Engage Amazon Q Developer in chat applications**: Integrate Amazon Q Developer in chat applications to relay alerts to Amazon Chime, Microsoft Teams, and Slack.
8. **Alert based on logs**: Use log metric filters in CloudWatch to create alarms based on specific log events.
9. **Review and iterate**: Regularly revisit and refine alert configurations.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS04-BP03 Implement user experience telemetry
- OPS04-BP04 Implement dependency telemetry
- OPS04-BP05 Implement distributed tracing
- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs
- OPS08-BP03 Analyze workload traces

Related documents:

- Using Amazon CloudWatch alarms
- Create a composite alarm
- Create a CloudWatch alarm based on anomaly detection
- DevOps Guru Notifications
- X-ray insights notifications
- Monitor, operate, and troubleshoot your AWS resources with interactive ChatOps
- Amazon CloudWatch Integration Guide | PagerDuty
- Integrate Opsgenie with Amazon CloudWatch

Related videos:

- Create Composite Alarms in Amazon CloudWatch
- Amazon Q Developer in chat applications Overview
- AWS On Air ft. Mutative Commands in Amazon Q Developer in chat applications

Related examples:

- Alarms, incident management, and remediation in the cloud with Amazon CloudWatch
- Tutorial: Creating an Amazon EventBridge rule that sends notifications to Amazon Q Developer in chat applications
- One Observability Workshop

OPS08-BP05 Create dashboards

Dashboards are the human-centric view into the telemetry data of your workloads. While they provide a vital visual interface, they should not replace alerting mechanisms, but complement them. When crafted with care, not only can they offer rapid insights into system health and performance, but they can also present stakeholders with real-time information on business outcomes and the impact of issues.

Desired outcome:

Clear, actionable insights into system and business health using visual representations.

Common anti-patterns:

- Overcomplicating dashboards with too many metrics.
- Relying on dashboards without alerts for anomaly detection.
- Not updating dashboards as workloads evolve.

Benefits of this best practice:

- Immediate visibility into critical system metrics and KPIs.
- Enhanced stakeholder communication and understanding.
- Rapid insight into the impact of operational issues.

Level of risk if this best practice isn't established: Medium

Implementation guidance

Business-centric dashboards

Dashboards tailored to business KPIs engage a wider array of stakeholders. While these individuals might not be interested in system metrics, they are keen on understanding the business implications of these numbers. A business-centric dashboard ensures that all technical and operational metrics being monitored and analyzed are in sync with overarching business goals. This alignment provides clarity, ensuring everyone is on the same page regarding what's essential and what's not. Additionally, dashboards that highlight business KPIs tend to be more actionable. Stakeholders can quickly understand the health of operations, areas that need attention, and the potential impact on business outcomes.

With this in mind, when creating your dashboards, ensure that there's a balance between technical metrics and business KPIs. Both are vital, but they cater to different audiences. Ideally, you should have dashboards that provide a holistic view of the system's health and performance while also emphasizing key business outcomes and their implications.

Amazon CloudWatch Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view, even those resources that are spread across different AWS Regions and accounts.

Implementation steps

1. **Create a basic dashboard:** Create a new dashboard in CloudWatch, giving it a descriptive name.

2. **Use Markdown widgets:** Before diving into the metrics, use Markdown widgets to add textual context at the top of your dashboard. This should explain what the dashboard covers, the significance of the represented metrics, and can also contain links to other dashboards and troubleshooting tools.
3. **Create dashboard variables:** Incorporate dashboard variables where appropriate to allow for dynamic and flexible dashboard views.
4. **Create metrics widgets:** Add metric widgets to visualize various metrics your application emits, tailoring these widgets to effectively represent system health and business outcomes.
5. **Log Insights queries:** Utilize CloudWatch Log Insights to derive actionable metrics from your logs and display these insights on your dashboard.
6. **Set up alarms:** Integrate CloudWatch Alarms into your dashboard for a quick view of any metrics breaching their thresholds.
7. **Use Contributor Insights:** Incorporate CloudWatch Contributor Insights to analyze high-cardinality fields and get a clearer understanding of your resource's top contributors.
8. **Design custom widgets:** For specific needs not met by standard widgets, consider creating custom widgets. These can pull from various data sources or represent data in unique ways.
9. **Use AWS Health:** AWS Health is the authoritative source of information about the health of your AWS Cloud resources. Use AWS Health Dashboard out of the box, or use AWS Health data in your own dashboards and tools so you have the right information available to make informed decisions.
10. **Iterate and refine:** As your application evolves, regularly revisit your dashboard to ensure its relevance.

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS08-BP01 Analyze workload metrics
- OPS08-BP02 Analyze workload logs
- OPS08-BP03 Analyze workload traces
- OPS08-BP04 Create actionable alerts

Related documents:

- Building Dashboards for Operational Visibility
- Using Amazon CloudWatch Dashboards

Related videos:

- Create Cross Account & Cross Region CloudWatch Dashboards
- AWS re:Invent 2021 - Gain enterprise visibility with AWS Cloud operation dashboards)

Related examples:

- One Observability Workshop
- Application Monitoring with Amazon CloudWatch
- AWS Health Events Intelligence Dashboards and Insights
- Visualize AWS Health events using Amazon Managed Grafana

OPS 9. How do you understand the health of your operations?

Define, capture, and analyze operations metrics to gain visibility to operations events so that you can take appropriate action.

Best practices

- OPS09-BP01 Measure operations goals and KPIs with metrics
- OPS09-BP02 Communicate status and trends to ensure visibility into operation
- OPS09-BP03 Review operations metrics and prioritize improvement

OPS09-BP01 Measure operations goals and KPIs with metrics

Obtain goals and KPIs that define operations success from your organization and determine that metrics reflect these. Set baselines as a point of reference and reevaluate regularly. Develop mechanisms to collect these metrics from teams for evaluation. The DevOps Research and Assessment (DORA) metrics provide a popular method to measure progress towards DevOps practices of software delivery.

Desired outcome:

- The organization publishes and shares the goals and KPIs for the operations teams.

- You establish metrics that reflect these KPIs. Examples may include:
  - Ticket queue depth or average age of ticket
  - Ticket count grouped by type of issue
  - Time spent working issues with or without a standardized operating procedure (SOP)
  - Amount of time spent recovering from a failed code push
  - Call volume

Common anti-patterns:

- Deployment deadlines are missed because developers are pulled away to perform troubleshooting tasks. Development teams argue for more personnel, but cannot quantify how many they need because the time taken away cannot be measured.
- A Tier 1 desk was set up to handle user calls. Over time, more workloads were added, but no headcount was allocated to the Tier 1 desk. Customer satisfaction suffers as call times increase and issues go longer without resolution, but management sees no indicators of such, preventing any action.
- A problematic workload has been handed off to a separate operations team for upkeep. Unlike other workloads, this new one was not supplied with proper documentation and runbooks. As such, teams spend longer troubleshooting and addressing failures. However, there are no metrics documenting this, which makes accountability difficult.

Benefits of establishing this best practice: Where workload monitoring shows the state of our applications and services, monitoring operations teams provides owners insight into changes among the consumers of those workloads, such as shifting business needs. Measure the effectiveness of these teams and evaluate them against business goals by creating metrics that can reflect the state of operations. Metrics can highlight support issues or identify when drifts occur away from a service level target.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Schedule time with business leaders and stakeholders to determine what the overall goals of the service will be. Determine what the tasks of various operations teams should be and what challenges they could be approached with. Using these, brainstorm key performance indicators

(KPIs) that might reflect these operations goals. These might be customer satisfaction, time from feature conception to deployment, average issue resolution time, or cost efficiencies.

Working from KPIs, identify the metrics and sources of data that might reflect these goals best. Customer satisfaction may be an combination of various metrics such as call wait or response times, satisfaction scores, and types of issues raised. Deployment times may be the sum of time needed for testing and deployment, plus any post-deployment fixes that needed to be added. Statistics showing the time spent on different types of issues (or the counts of those issues) can provide a window into where targeted effort is needed.

Resources

Related documents:

- Quick - Using KPIs
- Amazon CloudWatch - Using Metrics
- Building Dashboards
- How to track your cost optimization KPIs with KPI Dashboard
- AWS DevOps Guidance

Related examples:

- Monitor the performance of your software delivery using native AWS monitoring and observability tools
- Balance deployment speed and stability with DORA metrics
- Example MLOps operational metrics in the financial services industry
- How to track your cost optimization KPIs with the KPI Dashboard

OPS09-BP02 Communicate status and trends to ensure visibility into operation

Knowing the state of your operations and its trending direction is necessary to identify when outcomes may be at risk, whether or not added work can be supported, or the effects that changes have had to your teams. During operations events, having status pages that users and operations teams can refer to for information can reduce pressure on communication channels and disseminate information proactively.

Desired outcome:

- Operations leaders have insight at a glance to see what sort of call volumes their teams are operating under and what efforts may be under way, such as deployments.
- Alerts are disseminated to stakeholders and user communities when impacts to normal operations occur.
- Organization leadership and stakeholders can check a status page in response to an alert or impact, and obtain information surrounding an operational event, such as points of contact, ticket information, and estimated recovery times.
- Reports are made available to leadership and other stakeholders to show operations statistics such as call volumes over a period of time, user satisfaction scores, numbers of outstanding tickets and their ages.

Common anti-patterns:

- A workload goes down, leaving a service unavailable. Call volumes spike as users request to know what's going on. Managers add to the volume requesting to know who's working an issue. Various operations teams duplicate efforts in trying to investigate.
- A desire for a new capability leads to several personnel being reassigned to an engineering effort. No backfill is provided, and issue resolution times spike. This information is not captured, and only after several weeks and dissatisfied user feedback does leadership become aware of the issue.

Benefits of establishing this best practice: During operational events where the business is impacted, much time and energy can be wasted querying information from various teams attempting to understand the situation. By establishing widely-disseminated status pages and dashboards, stakeholders can quickly obtain information such as whether or not an issue was detected, who has lead on the issue, or when a return to normal operations may be expected. This frees team members from spending too much time communicating status to others and more time addressing issues.

In addition, dashboards and reports can provide insights to decision-makers and stakeholders to see how operations teams are able to respond to business needs and how their resources are being allocated. This is crucial for determining if adequate resources are in place to support the business.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Build dashboards that show the current key metrics for your ops teams, and make them readily accessible both to operations leaders and management.

Build status pages that can be updated quickly to show when an incident or event is unfolding, who has ownership and who is coordinating the response. Share any steps or workarounds that users should consider on this page, and disseminate the location widely. Encourage users to check this location first when confronted with an unknown issue.

Collect and provide reports that show the health of operations over time, and distribute this to leaders and decision makers to illustrate the work of operations along with challenges and needs.

Share between teams these metrics and reports that best reflect goals and KPIs and where they have been influential in driving change. Dedicate time to these activities to elevate the importance of operations inside of and between teams.

Use AWS Health alongside your own dashboards, or integrate AWS Health events into them, so that your teams can correlate application issues to AWS service status.

Resources

Related best practices:

- OPS09-BP01 Measure operations goals and KPIs with metrics

Related documents:

- Measure Progress
- Building dashboards for operation visibility

Related examples:

- Data Operations
- How to track your cost optimization KPIs with KPI Dashboard
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations

OPS09-BP03 Review operations metrics and prioritize improvement

Setting aside dedicated time and resources for reviewing the state of operations ensures that serving the day-to-day line of business remains a priority. Pull together operations leaders and stakeholders to regularly review metrics, reaffirm or modify goals and objectives, and prioritize improvements.

Desired outcome:

- Operations leaders and staff regularly meet to review metrics over a given reporting period. Challenges are communicated, wins are celebrated, and lessons learned are shared.
- Stakeholders and business leaders are regularly briefed on the state of operations and solicited for input regarding goals, KPIs, and future initiatives. Tradeoffs between service delivery, operations, and maintenance are discussed and placed into context.

Common anti-patterns:

- A new product is launched, but the Tier 1 and Tier 2 operations teams are not adequately trained to support or given additional staff. Metrics that show the decrease in ticket resolution times and increase in incident volumes are not seen by leaders. Action is taken weeks later when subscription numbers start to fall as discontent users move off the platform.
- A manual process for performing maintenance on a workload has been in place for a long time. While a desire to automate has been present, this was a low priority given the low importance of the system. Over time however, the system has grown in importance and now these manual processes consume a majority of operations' time. No resources are scheduled for providing increased tooling to operations, leading to staff burnout as workloads increase. Leadership becomes aware once it's reported that staff are leaving for other competitors.

Benefits of establishing this best practice: In some organizations, it can become a challenge to allocate the same time and attention that is afforded to service delivery and new products or offerings. When this occurs, the line of business can suffer as the level of service expected slowly deteriorates. This is because operations does not change and evolve with the growing business, and can soon be left behind. Without regular review into the insights operations collects, the risk to the business may become visible only when it's too late. By allocating time to review metrics and procedures both among the operations staff and with leadership, the crucial role operations plays remains visible, and risks can be identified long before they reach critical levels. Operations teams get better insight into impending business changes and initiatives, allowing for proactive

efforts to be undertaken. Leadership visibility into operations metrics showcases the role that these teams play in customer satisfaction, both internal and external, and let them better weigh choices for priorities, or ensure that operations has the time and resources to change and evolve with new business and workload initiatives.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Dedicate time to review operations metrics between stakeholders and operations teams and review report data. Place these reports in the contexts of the organizations goals and objectives to determine if they're being met. Identify sources of ambiguity where goals are not clear, or where there may be conflicts between what is asked for and what is given.

Identify where time, people, and tools can aid in operations outcomes. Determine which KPIs this would impact and what targets for success should be. Revisit regularly to ensure operations is resourced sufficiently to support the line of business.

Resources

Related documents:

- Amazon Athena
- Amazon CloudWatch metrics and dimensions reference
- Amazon Quick
- AWS Glue
- AWS Glue Data Catalog
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the Amazon CloudWatch Agent
- Using Amazon CloudWatch metrics

OPS 10. How do you manage workload and operations events?

Prepare and validate procedures for responding to events to minimize their disruption to your workload.

Best practices

- OPS10-BP01 Use a process for event, incident, and problem management
- OPS10-BP02 Have a process per alert
- OPS10-BP03 Prioritize operational events based on business impact
- OPS10-BP04 Define escalation paths
- OPS10-BP05 Define a customer communication plan for service-impacting events
- OPS10-BP06 Communicate status through dashboards
- OPS10-BP07 Automate responses to events

OPS10-BP01 Use a process for event, incident, and problem management

The ability to efficiently manage events, incidents, and problems is key to maintaining workload health and performance. It's crucial to recognize and understand the differences between these elements to develop an effective response and resolution strategy. Establishing and following a well-defined process for each aspect helps your team swiftly and effectively handle any operational challenges that arise.

Desired outcome: Your organization effectively manages operational events, incidents, and problems through well-documented and centrally stored processes. These processes are consistently updated to reflect changes, streamlining handling and maintaining high service reliability and workload performance.

Common anti-patterns:

- You reactively, rather than proactively, respond to events.
- Inconsistent approaches are taken to different types of events or incidents.
- Your organization does not analyze and learn from incidents to prevent future occurrences.

Benefits of establishing this best practice:

- Streamlined and standardized response processes.
- Reduced impact of incidents on services and customers.
- Expedited issue resolution.
- Continuous improvement in operational processes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing this best practice means you are tracking workload events. You have processes to handle incidents and problems. The processes are documented, shared, and updated frequently. Problems are identified, prioritized, and fixed.

Understanding events, incidents, and problems

- **Events:** An *event* is an observation of an action, occurrence, or change of state. Events can be planned or unplanned and they can originate internally or externally to the workload.
- **Incidents:** *Incidents* are events that require a response, like unplanned interruptions or degradations of service quality. They represent disruptions that need immediate attention to restore normal workload operation.
- **Problems:** *Problems* are the underlying causes of one or more incidents. Identifying and resolving problems involves digging deeper into the incidents to prevent future occurrences.

Implementation steps

Events

1. **Monitor events:**
   - Implement observability and utilize workload observability.
   - Monitor actions taken by a user, role, or an AWS service are recorded as events in AWS CloudTrail.
   - Respond to operational changes in your applications in real time with Amazon EventBridge.
   - Continually assess, monitor, and record resource configuration changes with AWS Config.
2. **Create processes:**
   - Develop a process to assess which events are significant and require monitoring. This involves setting thresholds and parameters for normal and abnormal activities.
   - Determine criteria escalating an event to an incident. This could be based on the severity, impact on users, or deviation from expected behavior.
   - Regularly review the event monitoring and response processes. This includes analyzing past incidents, adjusting thresholds, and refining alerting mechanisms.

Incidents

1. **Respond to incidents:**
    - Use insights from observability tools to quickly identify and respond to incidents.
    - Implement AWS Systems Manager Ops Center to aggregate, organize, and prioritize operational items and incidents.
    - Use services like Amazon CloudWatch and AWS X-Ray for deeper analysis and troubleshooting.
    - Consider AWS Managed Services (AMS) for enhanced incident management, leveraging its proactive, preventative, and detective capabilities. AMS extends operational support with services like monitoring, incident detection and response, and security management.
    - Enterprise Support customers can use AWS Incident Detection and Response, which provides continual proactive monitoring and incident management for production workloads.
2. **Create an incident management process:**
    - Establish a structured incident management process, including clear roles, communication protocols, and steps for resolution.
    - Integrate incident management with tools like Amazon Q Developer in chat applications for efficient response and coordination.
    - Categorize incidents by severity, with predefined incident response plans for each category.
3. **Learn and improve:**
    - Conduct post-incident analysis to understand root causes and resolution effectiveness.
    - Continually update and improve response plans based on reviews and evolving practices.
    - Document and share lessons learned across teams to enhance operational resilience.
    - Enterprise Support customers can request the Incident Management Workshop from their Technical Account Manager. This guided workshop tests your existing incident response plan and helps you identify areas for improvement.

Problems

1. **Identify problems:**
    - Use data from previous incidents to identify recurring patterns that may indicate deeper systemic issues.
    - Leverage tools like AWS CloudTrail and Amazon CloudWatch to analyze trends and uncover underlying problems.

- Engage cross-functional teams, including operations, development, and business units, to gain diverse perspectives on the root causes.

2. **Create a problem management process:**
   - Develop a structured process for problem management, focusing on long-term solutions rather than quick fixes.
   - Incorporate root cause analysis (RCA) techniques to investigate and understand the underlying causes of incidents.
   - Update operational policies, procedures, and infrastructure based on findings to prevent recurrence.
3. **Continue to improve:**
   - Foster a culture of constant learning and improvement, encouraging teams to proactively identify and address potential problems.
   - Regularly review and revise problem management processes and tools to align with evolving business and technology landscapes.
   - Share insights and best practices across the organization to build a more resilient and efficient operational environment.
4. **Engage AWS Support:**
   - Use AWS support resources, such as AWS Trusted Advisor, for proactive guidance and optimization recommendations.
   - Enterprise Support customers can access specialized programs like AWS Countdown for support during critical events.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS04-BP02 Implement application telemetry
- OPS07-BP03 Use runbooks to perform procedures
- OPS07-BP04 Use playbooks to investigate issues
- OPS08-BP01 Analyze workload metrics
- OPS11-BP02 Perform post-incident analysis

Related documents:

- AWS Security Incident Response Guide
- AWS Incident Detection and Response
- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- Incident Management in the Age of DevOps and SRE
- PagerDuty - What is Incident Management?

Related videos:

- Top incident response tips from AWS
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 yrs of Amazon operational excellence
- AWS re:Invent 2022 - AWS Incident Detection and Response (SUP201)
- Introducing Incident Manager from AWS Systems Manager

Related examples:

- AWS Proactive Services – Incident Management Workshop
- How to Automate Incident Response with PagerDuty and AWS Systems Manager Incident Manager
- Engage Incident Responders with the On-Call Schedules in AWS Systems Manager Incident Manager
- Improve the Visibility and Collaboration during Incident Handling in AWS Systems Manager Incident Manager
- Incident reports and service requests in AMS

Related services:

- Amazon EventBridge

OPS10-BP02 Have a process per alert

Establishing a clear and defined process for each alert in your system is essential for effective and efficient incident management. This practice ensures that every alert leads to a specific, actionable response, improving the reliability and responsiveness of your operations.

Desired outcome: Every alert initiates a specific, well-defined response plan. Where possible, responses are automated, with clear ownership and a defined escalation path. Alerts are linked to an up-to-date knowledge base so that any operator can respond consistently and effectively. Responses are quick and uniform across the board, enhancing operational efficiency and reliability.

Common anti-patterns:

- Alerts have no predefined response process, leading to makeshift and delayed resolutions.
- Alert overload causes important alerts to be overlooked.
- Alerts are inconsistently handled due to lack of clear ownership and responsibility.

Benefits of establishing this best practice:

- Reduced alert fatigue by only raising actionable alerts.
- Decreased mean time to resolution (MTTR) for operational issues.
- Decreased mean time to investigate (MTTI), which helps reduce MTTR.
- Enhanced ability to scale operational responses.
- Improved consistency and reliability in handling operational events.

For example, you have a defined process for AWS Health events for critical accounts, including application alarms, operational issues, and planned lifecycle events (like updating Amazon EKS versions before clusters are auto-updated), and you provide the capability for your teams to actively monitor, communicate, and respond to these events. These actions help you prevent service disruptions caused by AWS-side changes or mitigate them faster when unexpected issues occur.

Level of risk exposed if this best practice is not established: High

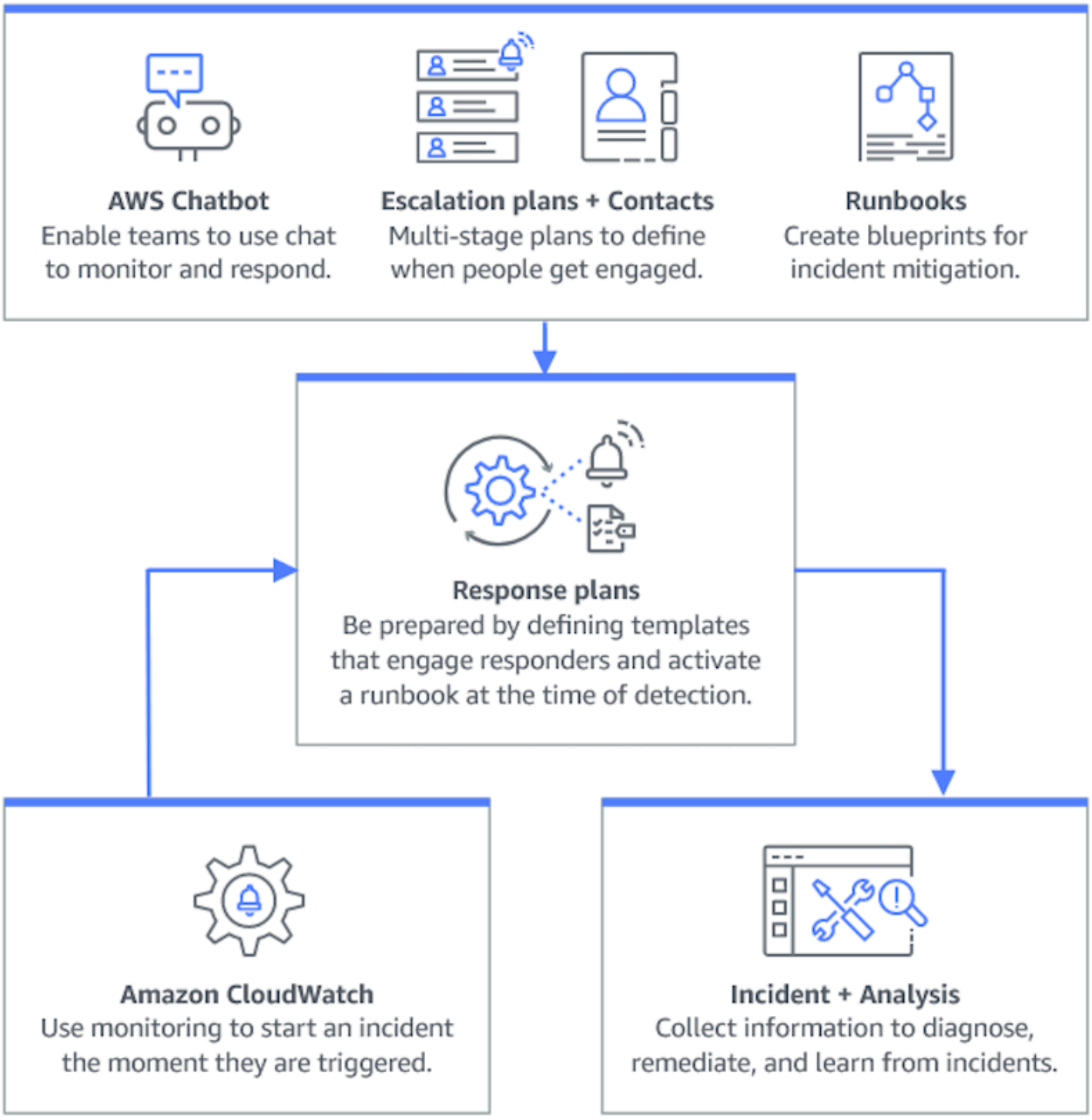
Implementation guidance

Having a process per alert involves establishing a clear response plan for each alert, automating responses where possible, and continually refining these processes based on operational feedback and evolving requirements.

Implementation steps

The following diagram illustrates the incident management workflow within AWS Systems Manager Incident Manager. It is designed to respond swiftly to operational issues by automatically

creating incidents in response to specific events from Amazon CloudWatch or Amazon EventBridge. When an incident is created, either automatically or manually, Incident Manager centralizes the management of the incident, organizes relevant AWS resource information, and initiates predefined response plans. This includes running Systems Manager Automation runbooks for immediate action, as well as creating a parent operational work item in OpsCenter to track related tasks and analyses. This streamlined process speeds up and coordinates incident response across your AWS environment.



1. **Use composite alarms:** Create composite alarms in CloudWatch to group related alarms, reducing noise and allowing for more meaningful responses.
2. **Stay informed with AWS Health:** AWS Health is the authoritative source of information about the health of your AWS Cloud resources. Use AWS Health to visualize and get notified of any

current service events and upcoming changes, such as planned lifecycle events, so you can take steps to mitigate impacts.

   a. Create purpose-fit AWS Health event notifications to e-mail and chat channels through AWS User Notifications, and integrate programatically with your monitoring and alerting tools through Amazon EventBridge or the AWS Health API.

   b. Plan and track progress on health events that require action by integrating with change management or ITSM tools (like Jira or ServiceNow) that you may already use through Amazon EventBridge or the AWS Health API.

   c. If you use AWS Organizations, enable organization view for AWS Health to aggregate AWS Health events across accounts.

3. **Integrate Amazon CloudWatch alarms with Incident Manager:** Configure CloudWatch alarms to automatically create incidents in AWS Systems Manager Incident Manager.
4. **Integrate Amazon EventBridge with Incident Manager:** Create EventBridge rules to react to events and create incidents using defined response plans.
5. **Prepare for incidents in Incident Manager:**
   - Establish detailed response plans in Incident Manager for each type of alert.
   - Establish chat channels through Amazon Q Developer in chat applications connected to response plans in Incident Manager, facilitating real-time communication during incidents across platforms like Slack, Microsoft Teams, and Amazon Chime.
   - Incorporate Systems Manager Automation runbooks within Incident Manager to drive automated responses to incidents.

Resources

Related best practices:

- OPS04-BP01 Identify key performance indicators
- OPS08-BP04 Create actionable alerts

Related documents:

- AWS Cloud Adoption Framework: Operations Perspective - Incident and problem management
- Using Amazon CloudWatch alarms
- Setting up AWS Systems Manager Incident Manager

- Preparing for incidents in Incident Manager

Related videos:

- Top incident response tips from AWS
- re:Invent 2023 | Manage resource lifecycle events at scale with AWS Health

Related examples:

- AWS Workshops - AWS Systems Manager Incident Manager - Automate incident response to security events

OPS10-BP03 Prioritize operational events based on business impact

Responding promptly to operational events is critical, but not all events are equal. When you prioritize based on business impact, you also prioritize addressing events with the potential for significant consequences, such as safety, financial loss, regulatory violations, or damage to reputation.

Desired outcome: Responses to operational events are prioritized based on potential impact to business operations and objectives. This makes the responses efficient and effective.

Common anti-patterns:

- Every event is treated with the same level of urgency, leading to confusion and delays in addressing critical issues.
- You fail to distinguish between high and low impact events, leading to misallocation of resources.
- Your organization lacks a clear prioritization framework, resulting in inconsistent responses to operational events.
- Events are prioritized based on the order they are reported, rather than their impact on business outcomes.

Benefits of establishing this best practice:

- Ensures critical business functions receive attention first, minimizing potential damage.
- Improves resource allocation during multiple concurrent events.

- Enhances the organization's ability to maintain trust and meet regulatory requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When faced with multiple operational events, a structured approach to prioritization based on impact and urgency is essential. This approach helps you make informed decisions, direct efforts where they're needed most, and mitigate the risk to business continuity.

Implementation steps

1. **Assess impact:** Develop a classification system to evaluate the severity of events in terms of their potential impact on business operations and objectives. The following example shows impact categories:

Impact level	Description
High	Affects many staff or customers, high financial impact, high reputational damage, or injury.
Medium	Affects a groups of staff or customers, moderate financial impact, or moderate reputational damage.
Low	Affects individual staff or customers, low financial impact, or low reputational damage.

2. **Assess urgency:** Define urgency levels for how quickly an event needs a response, considering factors such as safety, financial implications, and service-level agreements (SLAs). The following example demonstrates urgency categories:

Urgency level	Description
High	Exponentially increasing damage, time-sens itive work impacted, imminent escalation, or VIP users or groups affected.

Urgency level	Description
Medium	Damage increases over time, or single VIP user or group affected.
Low	Marginal damage increase over time, or non-time-sensitive work impacted.

3. **Create a prioritization matrix:**
    - Use a matrix to cross-reference impact and urgency, assigning priority levels to different combinations.
    - Make the matrix accessible and understood by all team members responsible for operational event responses.
    - The following example matrix displays incident severity according to urgency and impact:

Urgency and impact	High	Medium	Low
High	Critical	Urgent	High
Medium	Urgent	High	Normal
Low	High	Normal	Low

4. **Train and communicate:** Train response teams on the prioritization matrix and the importance of following it during an event. Communicate the prioritization process to all stakeholders to set clear expectations.
5. **Integrate with incident response:**
    - Incorporate the prioritization matrix into your incident response plans and tools.
    - Automate the classification and prioritization of events where possible to speed up response times.
    - Enterprise Support customers can leverage AWS Incident Detection and Response, which provides 24x7 proactive monitoring and incident management for production workloads.
6. **Review and adapt:** Regularly review the effectiveness of the prioritization process and make adjustments based on feedback and changes in the business environment.

Resources

Related best practices:

- OPS03-BP03 Escalation is encouraged
- OPS08-BP04 Create actionable alerts
- OPS09-BP01 Measure operations goals and KPIs with metrics

Related documents:

- Atlassian - Understanding incident severity levels
- IT Process Map - Checklist Incident Priority

OPS10-BP04 Define escalation paths

Establish clear escalation paths within your incident response protocols to facilitate timely and effective action. This includes specifying prompts for escalation, detailing the escalation process, and pre-approving actions to expedite decision-making and reduce mean time to resolution (MTTR).

Desired outcome: A structured and efficient process that escalates incidents to the appropriate personnel, minimizing response times and impact.

Common anti-patterns:

- Lack of clarity on recovery procedures leads to makeshift responses during critical incidents.
- Absence of defined permissions and ownership results in delays when urgent action is needed.
- Stakeholders and customers are not informed in line with expectations.
- Important decisions are delayed.

Benefits of establishing this best practice:

- Streamlined incident response through predefined escalation procedures.
- Reduced downtime with pre-approved actions and clear ownership.
- Improved resource allocation and support-level adjustments according to incident severity.
- Improved communication to stakeholders and customers.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Properly defined escalation paths are crucial for rapid incident response. AWS Systems Manager Incident Manager supports the setup of structured escalation plans and on-call schedules, which alert the right personnel so that they are ready to act when incidents occur.

Implementation steps

1. **Set up escalation prompts:** Set up CloudWatch alarms to create an incident in AWS Systems Manager Incident Manager.
2. **Set up on-call schedules:** Create on-call schedules in Incident Manager that align with your escalation paths. Equip on-call personnel with the necessary permissions and tools to act swiftly.
3. **Detail escalation procedures:**
    - Determine specific conditions under which an incident should be escalated.
    - Create escalation plans in Incident Manager.
    - Escalation channels should consist of a contact or an on-call schedule.
    - Define the roles and responsibilities of the team at each escalation level.
4. **Pre-approve mitigation actions:** Collaborate with decision-makers to pre-approve actions for anticipated scenarios. Use Systems Manager Automation runbooks integrated with Incident Manager to speed up incident resolution.
5. **Specify ownership:** Clearly identify internal owners for each step of the escalation path.
6. **Detail third-party escalations:**
    - Document third-party service-level agreements (SLAs), and align them with internal goals.
    - Set clear protocols for vendor communication during incidents.
    - Integrate vendor contacts into incident management tools for direct access.
    - Conduct regular drills that include third-party response scenarios.
    - Keep vendor escalation information well-documented and easily accessible.
7. **Train and rehearse escalation plans:** Train your team on the escalation process and conduct regular incident response drills or game days. Enterprise Support customers can request an Incident Management Workshop.
8. **Continue to improve:** Review the effectiveness of your escalation paths regularly. Update your processes based on lessons learned from incident post-mortems and continuous feedback.

Level of effort for the implementation plan: Moderate

Resources

Related best practices:

- OPS08-BP04 Create actionable alerts
- OPS10-BP02 Have a process per alert
- OPS11-BP02 Perform post-incident analysis

Related documents:

- AWS Systems Manager Incident Manager Escalation Plans
- Working with on-call schedules in Incident Manager
- Creating and Managing Runbooks
- Temporary elevated access management with AWS IAM Identity Center
- Atlassian - Escalation policies for effective incident management

OPS10-BP05 Define a customer communication plan for service-impacting events

Effective communication during service impacting events is critical to maintain trust and transparency with customers. A well-defined communication plan helps your organization quickly and clearly share information, both internally and externally, during incidents.

Desired outcome:

- A robust communication plan that effectively informs customers and stakeholders during service impacting events.
- Transparency in communication to build trust and reduce customer anxiety.
- Minimizing the impact of service impacting events on customer experience and business operations.

Common anti-patterns:

- Inadequate or delayed communication leads to customer confusion and dissatisfaction.
- Overly technical or vague messaging fails to convey the actual impact on users.
- There is no predefined communication strategy, resulting in inconsistent and reactive messaging.

Benefits of establishing this best practice:

- Enhanced customer trust and satisfaction through proactive and clear communication.
- Reduced burden on support teams by preemptively addressing customer concerns.
- Improved ability to manage and recover from incidents effectively.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Creating a comprehensive communication plan for service impacting events involves multiple facets, from choosing the right channels to crafting the message and tone. The plan should be adaptable, scalable, and cater to different outage scenarios.

Implementation steps

1. **Define roles and responsibilities:**
   - Assign a major incident manager to oversee incident response activities.
   - Designate a communications manager responsible for coordinating all external and internal communications.
   - Include the support manager to provide consistent communication through support tickets.
2. **Identify communication channels:** Select channels like workplace chat, email, SMS, social media, in-app notifications, and status pages. These channels should be resilient and able to operate independently during service impacting events.
3. **Communicate quickly, clearly, and regularly to customers:**
   - Develop templates for various service impairment scenarios, emphasizing simplicity and essential details. Include information about the service impairment, expected resolution time, and impact.
   - Use Amazon Pinpoint to alert customers using push notifications, in-app notifications, emails, text messages, voice messages, and messages over custom channels.
   - Use Amazon Simple Notification Service (Amazon SNS) to alert subscribers programatically or through email, mobile push notifications, and text messages.
   - Communicate status through dashboards by sharing an Amazon CloudWatch dashboard publicly.
   - Encourage social media engagement:
     - Actively monitor social media to understand customer sentiment.

    - Post on social media platforms for public updates and community engagement.
    - Prepare templates for consistent and clear social media communication.
4. **Coordinate internal communication:** Implement internal protocols using tools like Amazon Q Developer in chat applications for team coordination and communication. Use CloudWatch dashboards to communicate status.
5. **Orchestrate communication with dedicated tools and services:**
    - Use AWS Systems Manager Incident Manager with Amazon Q Developer in chat applications to set up dedicated chat channels for real-time internal communication and coordination during incidents.
    - Use AWS Systems Manager Incident Manager runbooks to automate customer notifications through Amazon Pinpoint, Amazon SNS, or third-party tools like social media platforms during incidents.
    - Incorporate approval workflows within runbooks to optionally review and authorize all external communications before sending.
6. **Practice and improve:**
    - Conduct training on the use of communication tools and strategies. Empower teams to make timely decisions during incidents.
    - Test the communication plan through regular drills or gamedays. Use these tests to refine messaging and evaluate the effectiveness of channels.
    - Implement feedback mechanisms to assess communication effectiveness during incidents. Continually evolve the communication plan based on feedback and changing needs.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS07-BP03 Use runbooks to perform procedures
- OPS10-BP06 Communicate status through dashboards
- OPS11-BP02 Perform post-incident analysis

Related documents:

- Atlassian - Incident communication best practices

- Atlassian - How to write a good status update
- PagerDuty - A Guide to Incident Communications

Related videos:

- Atlassian - Create your own incident communication plan: Incident templates

Related examples:

- AWS Health Dashboard

OPS10-BP06 Communicate status through dashboards

Use dashboards as a strategic tool to convey real-time operational status and key metrics to different audiences, including internal technical teams, leadership, and customers. These dashboards offer a centralized, visual representation of system health and business performance, enhancing transparency and decision-making efficiency.

Desired outcome:

- Your dashboards provide a comprehensive view of the system and business metrics relevant to different stakeholders.
- Stakeholders can proactively access operational information, reducing the need for frequent status requests.
- Real-time decision-making is enhanced during normal operations and incidents.

Common anti-patterns:

- Engineers joining an incident management call require status updates to get up to speed.
- Relying on manual reporting for management, which leads to delays and potential inaccuracies.
- Operations teams are frequently interrupted for status updates during incidents.

Benefits of establishing this best practice:

- Empowers stakeholders with immediate access to critical information, promoting informed decision-making.

- Reduces operational inefficiencies by minimizing manual reporting and frequent status inquiries.
- Increases transparency and trust through real-time visibility into system performance and business metrics.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Dashboards effectively communicate the status of your system and business metrics and can be tailored to the needs of different audience groups. Tools like Amazon CloudWatch dashboards and Amazon Quick help you create interactive, real-time dashboards for system monitoring and business intelligence.

Implementation steps

1. **Identify stakeholder needs:** Determine the specific information needs of different audience groups, such as technical teams, leadership, and customers.
2. **Choose the right tools:** Select appropriate tools like Amazon CloudWatch dashboards for system monitoring and Amazon Quick for interactive business intelligence. AWS Health provides a ready-to-use experience in the AWS Health Dashboard, or you can use Health events in Amazon EventBridge or through the AWS Health API to augment your own dashboards.
3. **Design effective dashboards:**
   - Design dashboards to clearly present relevant metrics and KPIs, ensuring they are understandable and actionable.
   - Incorporate system-level and business-level views as needed.
   - Include both high-level (for broad overviews) and low-level (for detailed analysis) dashboards.
   - Integrate automated alarms within dashboards to highlight critical issues.
   - Annotate dashboards with important metrics thresholds and goals for immediate visibility.
4. **Integrate data sources:**
   - Use Amazon CloudWatch to aggregate and display metrics from various AWS services and query metrics from other data sources, creating a unified view of your system's health and business metrics.
   - Use features like CloudWatch Logs Insights to query and visualize log data from different applications and services.

- Use AWS Health events to stay informed about the operational status and confirmed operational issues from AWS services through the AWS Health API or AWS Health events on Amazon EventBridge.

5. **Provide self-service access:**
   - Share CloudWatch dashboards with relevant stakeholders for self-service information access using dashboard sharing features.
   - Ensure that dashboards are easily accessible and provide real-time, up-to-date information.
6. **Regularly update and refine:**
   - Continually update and refine dashboards to align with evolving business needs and stakeholder feedback.
   - Regularly review the dashboards to keep them relevant and effective for conveying the necessary information.

Resources

Related best practices:

- OPS08-BP05 Create dashboards

Related documents:

- Building dashboards for operational visibility
- Using Amazon CloudWatch dashboards
- Create flexible dashboards with dashboard variables
- Sharing CloudWatch dashboards
- Query metrics from other data sources
- Add a custom widget to a CloudWatch dashboard

Related examples:

- One Observability Workshop - Dashboards

OPS10-BP07 Automate responses to events

Automating event responses is key for fast, consistent, and error-free operational handling. Create streamlined processes and use tools to automatically manage and respond to events, minimizing manual interventions and enhancing operational effectiveness.

Desired outcome:

- Reduced human errors and faster resolution times through automation.
- Consistent and reliable operational event handling.
- Enhanced operational efficiency and system reliability.

Common anti-patterns:

- Manual event handling leads to delays and errors.
- Automation is overlooked in repetitive, critical tasks.
- Repetitive, manual tasks lead to alert fatigue and missing critical issues.

Benefits of establishing this best practice:

- Accelerated event responses, reducing system downtime.
- Reliable operations with automated and consistent event handling.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Incorporate automation to create efficient operational workflows and minimize manual interventions.

Implementation steps

1. **Identify automation opportunites:** Determine repetitive tasks for automation, such as issue remediation, ticket enrichment, capacity management, scaling, deployments, and testing.
2. **Identify automation prompts:**
   - Assess and define specific conditions or metrics that initiate automated responses using Amazon CloudWatch alarm actions.

- Use Amazon EventBridge to respond to events in AWS services, custom workloads, and SaaS applications.
- Consider initiation events such as specific log entries, performance metrics thresholds, or state changes in AWS resources.

3. **Implement event-driven automation:**
   - Use AWS Systems Manager Automation runbooks to simplify maintenance, deployment, and remediation tasks.
   - Creating incidents in Incident Manager automatically gathers and adds details about the involved AWS resources to the incident.
   - Proactively monitor quotas using Quota Monitor for AWS.
   - Automatically adjust capacity with AWS Auto Scaling to maintain availability and performance.
   - Automate development pipelines with Amazon CodeCatalyst.
   - Smoke test or continually monitor endpoints and APIs using synthetic monitoring.
4. **Perform risk mitigation through automation:**
   - Implement automated security responses to swiftly address risks.
   - Use AWS Systems Manager State Manager to reduce configuration drift.
   - Remediate noncompliant resources with AWS Config Rules.

Level of effort for the implementation plan: High

Resources

Related best practices:

- OPS08-BP04 Create actionable alerts
- OPS10-BP02 Have a process per alert

Related documents:

- Using Systems Manager Automation runbooks with Incident Manager
- Creating incidents in Incident Manager
- AWS service quotas
- Monitor resource usage and send notifications when approaching quotas

- AWS Auto Scaling
- What is Amazon CodeCatalyst?
- Using Amazon CloudWatch alarms
- Using Amazon CloudWatch alarm actions
- Remediating Noncompliant Resources with AWS Config Rules
- Creating metrics from log events using filters
- AWS Systems Manager State Manager

Related videos:

- Create Automation Runbooks with AWS Systems Manager
- How to automate IT Operations on AWS
- AWS Security Hub CSPM automation rules
- Start your software project fast with Amazon CodeCatalyst blueprints

Related examples:

- Amazon CodeCatalyst Tutorial: Creating a project with the Modern three-tier web application blueprint
- One Observability Workshop
- Respond to incidents using Incident Manager

Evolve

Question

- OPS 11. How do you evolve operations?

OPS 11. How do you evolve operations?

Dedicate time and resources for nearly continuous incremental improvement to evolve the effectiveness and efficiency of your operations.

Best practices

- OPS11-BP01 Have a process for continuous improvement

- OPS11-BP02 Perform post-incident analysis
- OPS11-BP03 Implement feedback loops
- OPS11-BP04 Perform knowledge management
- OPS11-BP05 Define drivers for improvement
- OPS11-BP06 Validate insights
- OPS11-BP07 Perform operations metrics reviews
- OPS11-BP08 Document and share lessons learned
- OPS11-BP09 Allocate time to make improvements

OPS11-BP01 Have a process for continuous improvement

Evaluate your workload against internal and external architecture best practices. Conduct frequent, intentional workload reviews. Prioritize improvement opportunities into your software development cadence.

Desired outcome:

- You analyze your workload against architecture best practices frequently.
- You give improvement opportunities equal priority to features in your software development process.

Common anti-patterns:

- You have not conducted an architecture review on your workload since it was deployed several years ago.
- You give a lower priority to improvement opportunities. Compared to new features, these opportunities stay in the backlog.
- There is no standard for implementing modifications to best practices for the organization.

Benefits of establishing this best practice:

- Your workload is kept up-to-date on architecture best practices.
- You evolve your workload in an intentional manner.
- You can leverage organization best practices to improve all workloads.
- You make marginal gains that have a cumulative impact, which drives deeper efficiencies.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Frequently conduct an architectural review of your workload. Use internal and external best practices, evaluate your workload, and identify improvement opportunities. Prioritize improvement opportunities into your software development cadence.

Implementation steps

1. Conduct periodic architecture reviews of your production workload with an agreed-upon frequency. Use a documented architectural standard that includes AWS-specific best practices.
   a. Use your internally-defined standards for these reviews. If you do not have an internal standard, use the AWS Well-Architected Framework.
   b. Use the AWS Well-Architected Tool to create a custom lens of your internal best practices and conduct your architecture review.
   c. Contact your AWS Solution Architect or Technical Account Manager to conduct a guided Well-Architected Framework Review of your workload.
2. Prioritize improvement opportunities identified during the review into your software development process.

Level of effort for the implementation plan: Low. You can use the AWS Well-Architected Framework to conduct your yearly architecture review.

Resources

Related best practices:

- OPS11-BP02 Perform post-incident analysis
- OPS11-BP08 Document and share lessons learned
- OPS04 Implement Observability

Related documents:

- AWS Well-Architected Tool - Custom lenses
- AWS Well-Architected Whitepaper - The review process
- Customize Well-Architected Reviews using Custom Lenses and the AWS Well-Architected Tool

- Implementing the AWS Well-Architected Custom Lens lifecycle in your organization

Related videos:

- AWS re:Invent 2023 - Scaling AWS Well-Architected best practices across your organization

Related examples:

- AWS Well-Architected Tool

OPS11-BP02 Perform post-incident analysis

Review customer-impacting events and identify the contributing factors and preventative actions. Use this information to develop mitigations to limit or prevent recurrence. Develop procedures for prompt and effective responses. Communicate contributing factors and corrective actions as appropriate, tailored to target audiences.

Desired outcome:

- You have established incident management processes that include post-incident analysis.
- You have observability plans in place to collect data on events.
- With this data, you understand and collect metrics that support your post-incident analysis process.
- You learn from incidents to improve future outcomes.

Common anti-patterns:

- You administer an application server. Approximately every 23 hours and 55 minutes all your active sessions are terminated. You have tried to identify what is going wrong on your application server. You suspect it could instead be a network issue but are unable to get cooperation from the network team as they are too busy to support you. You lack a predefined process to follow to get support and collect the information necessary to determine what is going on.
- You have had data loss within your workload. This is the first time it has happened and the cause is not obvious. You decide it is not important because you can recreate the data. Data loss starts occurring with greater frequency impacting your customers. This also places addition operational burden on you as you restore the missing data.

Benefits of establishing this best practice:

- You have a predefined process to determine the components, conditions, actions, and events that contributed to an incident, which helps you identify opportunities for improvement.
- You use data from post-incident analysis to make improvements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Use a process to determine contributing factors. Review all customer impacting incidents. Have a process to identify and document the contributing factors of an incident so that you can develop mitigations to limit or prevent recurrence and you can develop procedures for prompt and effective responses. Communicate incident root causes as appropriate, and tailor the communication to your target audience. Share learnings openly within your organization.

Implementation steps

1. Collect metrics such as deployment change, configuration change, incident start time, alarm time, time of engagement, mitigation start time, and incident resolved time.
2. Describe key time points on the timeline to understand the events of the incident.
3. Ask the following questions:
    a. Could you improve time to detection?
    b. Are there updates to metrics and alarms that would detect the incident sooner?
    c. Can you improve the time to diagnosis?
    d. Are there updates to your response plans or escalation plans that would engage the correct responders sooner?
    e. Can you improve the time to mitigation?
    f. Are there runbook or playbook steps that you could add or improve?
    g. Can you prevent future incidents from occurring?
4. Create checklists and actions. Track and deliver all actions.

Level of effort for the implementation plan: Medium

Resources

Related best practices:

- OPS11-BP01 Have a process for continuous improvement
- OPS 4 - Implement observability

Related documents:

- Performing a post-incident analysis in Incident Manager
- Operational Readiness Review

OPS11-BP03 Implement feedback loops

Feedback loops provide actionable insights that drive decision making. Build feedback loops into your procedures and workloads. This helps you identify issues and areas that need improvement. They also validate investments made in improvements. These feedback loops are the foundation for continuously improving your workload.

Feedback loops fall into two categories: *immediate feedback* and *retrospective analysis*. Immediate feedback is gathered through review of the performance and outcomes from operations activities. This feedback comes from team members, customers, or the automated output of the activity. Immediate feedback is received from things like A/B testing and shipping new features, and it is essential to failing fast.

Retrospective analysis is performed regularly to capture feedback from the review of operational outcomes and metrics over time. These retrospectives happen at the end of a sprint, on a cadence, or after major releases or events. This type of feedback loop validates investments in operations or your workload. It helps you measure success and validates your strategy.

Desired outcome: You use immediate feedback and retrospective analysis to drive improvements. There is a mechanism to capture user and team member feedback. Retrospective analysis is used to identify trends that drive improvements.

Common anti-patterns:

- You launch a new feature but have no way of receiving customer feedback on it.
- After investing in operations improvements, you don’t conduct a retrospective to validate them.
- You collect customer feedback but don’t regularly review it.
- Feedback loops lead to proposed action items but they aren’t included in the software development process.

- Customers don't receive feedback on improvements they've proposed.

Benefits of establishing this best practice:

- You can work backwards from the customer to drive new features.
- Your organization culture can react to changes faster.
- Trends are used to identify improvement opportunities.
- Retrospectives validate investments made to your workload and operations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing this best practice means that you use both immediate feedback and retrospective analysis. These feedback loops drive improvements. There are many mechanisms for immediate feedback, including surveys, customer polls, or feedback forms. Your organization also uses retrospectives to identify improvement opportunities and validate initiatives.

Customer example

AnyCompany Retail created a web form where customers can give feedback or report issues. During the weekly scrum, user feedback is evaluated by the software development team. Feedback is regularly used to steer the evolution of their platform. They conduct a retrospective at the end of each sprint to identify items they want to improve.

Implementation steps

1. Immediate feedback
   - You need a mechanism to receive feedback from customers and team members. Your operations activities can also be configured to deliver automated feedback.
   - Your organization needs a process to review this feedback, determine what to improve, and schedule the improvement.
   - Feedback must be added into your software development process.
   - As you make improvements, follow up with the feedback submitter.
     - You can use AWS Systems Manager OpsCenter to create and track these improvements as OpsItems.
2. Retrospective analysis

- Conduct retrospectives at the end of a development cycle, on a set cadence, or after a major release.
- Gather stakeholders involved in the workload for a retrospective meeting.
- Create three columns on a whiteboard or spreadsheet: Stop, Start, and Keep.
  - *Stop* is for anything that you want your team to stop doing.
  - *Start* is for ideas that you want to start doing.
  - *Keep* is for items that you want to keep doing.
- Go around the room and gather feedback from the stakeholders.
- Prioritize the feedback. Assign actions and stakeholders to any Start or Keep items.
- Add the actions to your software development process and communicate status updates to stakeholders as you make the improvements.

Level of effort for the implementation plan: Medium. To implement this best practice, you need a way to take in immediate feedback and analyze it. Also, you need to establish a retrospective analysis process.

Resources

Related best practices:

- OPS01-BP01 Evaluate external customer needs: Feedback loops are a mechanism to gather external customer needs.
- OPS01-BP02 Evaluate internal customer needs: Internal stakeholders can use feedback loops to communicate needs and requirements.
- OPS11-BP02 Perform post-incident analysis: Post-incident analyses are an important form of retrospective analysis conducted after incidents.
- OPS11-BP07 Perform operations metrics reviews: Operations metrics reviews identify trends and areas for improvement.

Related documents:

- 7 Pitfalls to Avoid When Building a CCOE
- Atlassian Team Playbook - Retrospectives
- Email Definitions: Feedback Loops

- Establishing Feedback Loops Based on the AWS Well-Architected Framework Review
- IBM Garage Methodology - Hold a retrospective
- Investopedia – The PDCS Cycle
- Maximizing Developer Effectiveness by Tim Cochran
- Operations Readiness Reviews (ORR) Whitepaper - Iteration
- ITIL CSI - Continual Service Improvement
- When Toyota met e-commerce: Lean at Amazon

Related videos:

- Building Effective Customer Feedback Loops

Related examples:

- Astuto - Open source customer feedback tool
- AWS Solutions - QnABot on AWS
- Fider - A platform to organize customer feedback

Related services:

- AWS Systems Manager OpsCenter

OPS11-BP04 Perform knowledge management

Knowledge management helps team members find the information to perform their job. In learning organizations, information is freely shared which empowers individuals. The information can be discovered or searched. Information is accurate and up to date. Mechanisms exist to create new information, update existing information, and archive outdated information. The most common example of a knowledge management platform is a content management system like a wiki.

Desired outcome:

- Team members have access to timely, accurate information.
- Information is searchable.

- Mechanisms exist to add, update, and archive information.

Common anti-patterns:

- There is no centralized knowledge storage. Team members manage their own notes on their local machines.
- You have a self-hosted wiki but no mechanisms to manage information, resulting in outdated information.
- Someone identifies missing information but there's no process to request adding it the team wiki. They add it themselves but they miss a key step, leading to an outage.

Benefits of establishing this best practice:

- Team members are empowered because information is shared freely.
- New team members are onboarded faster because documentation is up to date and searchable.
- Information is timely, accurate, and actionable.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Knowledge management is an important facet of learning organizations. To begin, you need a central repository to store your knowledge (as a common example, a self-hosted wiki). You must develop processes for adding, updating, and archiving knowledge. Develop standards for what should be documented and let everyone contribute.

Customer example

AnyCompany Retail hosts an internal Wiki where all knowledge is stored. Team members are encouraged to add to the knowledge base as they go about their daily duties. On a quarterly basis, a cross-functional team evaluates which pages are least updated and determines if they should be archived or updated.

Implementation steps

1. Start with identifying the content management system where knowledge will be stored. Get agreement from stakeholders across your organization.

   a. If you don't have an existing content management system, consider running a self-hosted wiki or using a version control repository as a starting point.
2. Develop runbooks for adding, updating, and archiving information. Educate your team on these processes.
3. Identify what knowledge should be stored in the content management system. Start with daily activities (runbooks and playbooks) that team members perform. Work with stakeholders to prioritize what knowledge is added.
4. On a periodic basis, work with stakeholders to identify out-of-date information and archive it or bring it up to date.

Level of effort for the implementation plan: Medium. If you don't have an existing content management system, you can set up a self-hosted wiki or a version-controlled document repository.

Resources

Related best practices:

- OPS11-BP08 Document and share lessons learned - Knowledge management facilitates information sharing about lessons learned.

Related documents:

- Atlassian - Knowledge Management

Related examples:

- DokuWiki
- Gollum
- MediaWiki
- Wiki.js

OPS11-BP05 Define drivers for improvement

Identify drivers for improvement to help you evaluate and prioritize opportunities based on data and feedback loops. Explore improvement opportunities in your systems and processes, and automate where appropriate.

Desired outcome:

- You track data from across your environment.
- You correlate events and activities to business outcomes.
- You can compare and contrast between environments and systems.
- You maintain a detailed activity history of your deployments and outcomes.
- You collect data to support your security posture.

Common anti-patterns:

- You collect data from across your environment but do not correlate events and activities.
- You collect detailed data from across your estate, and it drives high Amazon CloudWatch and AWS CloudTrail activity and cost. However, you do not use this data meaningfully.
- You do not account for business outcomes when defining drivers for improvement.
- You do not measure the effects of new features.

Benefits of establishing this best practice:

- You minimize the impact of event-based motivations or emotional investment by determining criteria for improvement.
- You respond to business events, not just technical ones.
- You measure your environment to identify areas of improvement.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Understand drivers for improvement: You should only make changes to a system when a desired outcome is supported.

- Desired capabilities: Evaluate desired features and capabilities when evaluating opportunities for improvement.
  - What's New with AWS
- Unacceptable issues: Evaluate unacceptable issues, bugs, and vulnerabilities when evaluating opportunities for improvement. Track rightsizing options, and seek optimization opportunities.
  - AWS Latest Security Bulletins
  - AWS Trusted Advisor
  - Cloud Intelligence Dashboards
- Compliance requirements: Evaluate updates and changes required to maintain compliance with regulation, policy, or to remain under support from a third party, when reviewing opportunities for improvement.
  - AWS Compliance
  - AWS Compliance Programs
  - AWS Compliance Latest News

Resources

Related best practices:

- OPS01 Organization priorities
- OPS02 Relationships and Ownerships
- OPS04-BP01 Identify key performance indicators
- OPS08 Utilizing Workload Observability
- OPS09 Understanding Operational Health
- OPS11-BP03 Implement feedback loops

Related documents:

- Amazon Athena
- Quick
- AWS Compliance
- AWS Compliance Latest News
- AWS Compliance Programs

- AWS Glue
- AWS Latest Security Bulletins
- AWS Trusted Advisor
- Export your log data to Amazon S3
- What's New with AWS
- The Imperatives of Customer-Centric Innovation
- Digital Transformation: Hype or a Strategic Necessity?

Related Videos

- AWS re:Invent 2023 - Improve operational efficiency and resilience with Support (SUP310)

OPS11-BP06 Validate insights

Review your analysis results and responses with cross-functional teams and business owners. Use these reviews to establish common understanding, identify additional impacts, and determine courses of action. Adjust responses as appropriate.

Desired outcomes:

- You review insights with business owners on a regular basis. Business owners provide additional context to newly-gained insights.
- You review insights and request feedback from technical peers, and you share your learnings across teams.
- You publish data and insights for other technical and business teams to review. You factor in your learnings to new practices by other departments.
- Summarize and review new insights with senior leaders. Senior leaders use new insights to define strategy.

Common anti-patterns:

- You release a new feature. This feature changes some of your customer behaviors. Your observability does not take these changes into account. You do not quantify the benefits of these changes.

- You push a new update and neglect refreshing your CDN. The CDN cache is no longer compatible with the latest release. You measure the percentage of requests with errors. All of your users report HTTP 400 errors when communicating with backend servers. You investigate the client errors and find that because you measured the wrong dimension, your time was wasted.
- Your service-level agreement stipulates 99.9% uptime, and your recovery point objective is four hours. The service owner maintains that the system is zero downtime. You implement an expensive and complex replication solution, which wastes time and money.

Benefits of establishing this best practice:

- When you validate insights with business owners and subject matter experts, you establish common understanding and more effectively guide improvement.
- You discover hidden issues and factor them into future decisions.
- Your focus moves from technical outcomes to business outcomes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- **Validate insights:** Engage with business owners and subject matter experts to ensure there is common understanding and agreement of the meaning of the data you have collected. Identify additional concerns, potential impacts, and determine a courses of action.

Resources

Related best practices:

- OPS01-BP06 Evaluate tradeoffs while managing benefits and risks
- OPS02-BP06 Responsibilities between teams are predefined or negotiated
- OPS11-BP03 Implement feedback loops

Related documents:

- Designing a Cloud Center of Excellence (CCOE)

Related videos:

- Building observability to increase resiliency

OPS11-BP07 Perform operations metrics reviews

Regularly perform retrospective analysis of operations metrics with cross-team participants from different areas of the business. Use these reviews to identify opportunities for improvement, potential courses of action, and to share lessons learned. Look for opportunities to improve in all of your environments (for example, development, test, and production).

Desired outcome:

- You frequently review business-affecting metrics
- You detect and review anomalies through your observability capabilities
- You use data to support business outcomes and goals

Common anti-patterns:

- Your maintenance window interrupts a significant retail promotion. The business remains unaware that there is a standard maintenance window that could be delayed if there are other business impacting events.
- You suffered an extended outage because you commonly use an outdated library in your organization. You have since migrated to a supported library. The other teams in your organization do not know that they are at risk.
- You do not regularly review attainment of customer SLAs. You are trending to not meet your customer SLAs. There are financial penalties related to not meeting your customer SLAs.

Benefits of establishing this best practice:

- When you meet regularly to review operations metrics, events, and incidents, you maintain common understanding across teams.
- Your team meets routinely to review metrics and incidents, which positions you to take action on risks and recognize customer SLAs.
- You share lessons learned, which provides data for prioritization and targeted improvements for business outcomes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Regularly perform retrospective analysis of operations metrics with cross-team participants from different areas of the business.
- Engage stakeholders, including the business, development, and operations teams, to validate your findings from immediate feedback and retrospective analysis and share lessons learned.
- Use their insights to identify opportunities for improvement and potential courses of action.

Resources

Related best practices:

- OPS08-BP05 Create dashboards
- OPS09-BP03 Review operations metrics and prioritize improvement
- OPS10-BP01 Use a process for event, incident, and problem management

Related documents:

- Amazon CloudWatch
- Amazon CloudWatch metrics and dimensions reference
- Publish custom metrics
- Using Amazon CloudWatch metrics
- Dashboards and visualizations with CloudWatch

OPS11-BP08 Document and share lessons learned

Document and share lessons learned from the operations activities so that you can use them internally and across teams. You should share what your teams learn to increase the benefit across your organization. Share information and resources to prevent avoidable errors and ease development efforts, and focus on delivery of desired features.

Use AWS Identity and Access Management (IAM) to define permissions that permit controlled access to the resources you wish to share within and across accounts.

Desired outcome:

- You use version-controlled repositories to share application libraries, scripted procedures, procedure documentation, and other system documentation.
- You share your infrastructure standards as version-controlled AWS CloudFormation templates.
- You review lessons learned across teams.

Common anti-patterns:

- You suffered an extended outage because your organization commonly uses buggy library. You have since migrated to a reliable library. The other teams in your organization do not know they are at risk. No one documents and shares the experience with this library, and they are not aware of the risk.
- You have identified an edge case in an internally-shared microservice that causes sessions to drop. You have updated your calls to the service to avoid this edge case. The other teams in your organization do not know that they are at risk.
- You have found a way to significantly reduce the CPU utilization requirements for one of your microservices. You do not know if any other teams could take advantage of this technique.

Benefits of establishing this best practice: Share lessons learned to support improvement and to maximize the benefits of experience.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- **Document and share lessons learned:** Have procedures to document the lessons learned from the running of operations activities and retrospective analysis so that they can be used by other teams.
- **Share learnings:** Have procedures to share lessons learned and associated artifacts across teams. For example, share updated procedures, guidance, governance, and best practices through an accessible wiki. Share scripts, code, and libraries through a common repository.
  - Leverage AWS re:Post Private as a knowledge service to streamline collaboration and knowledge sharing within your organization.

Resources

Related best practices:

- OPS02-BP06 Responsibilities between teams are predefined or negotiated
- OPS05-BP01 Use version control
- OPS05-BP06 Share design standards
- OPS11-BP03 Implement feedback loops
- OPS11-BP07 Perform operations metrics reviews

Related documents:

- Increase collaboration and securely share cloud knowledge with AWS re:Post Private
- Reduce project delays with a docs-as-code solution

Related videos:

- AWS re:Invent 2023 - Collaborate within your company and with AWS using AWS re:Post Private
- Supports You | Exploring the Incident Management Tabletop Exercise

OPS11-BP09 Allocate time to make improvements

Dedicate time and resources within your processes to make continuous incremental improvements possible.

Desired outcome:

- You create temporary duplicates of environments, which lowers the risk, effort, and cost of experimentation and testing.
- These duplicated environments can be used to test the conclusions from your analysis, experiment, and develop and test planned improvements.
- You run gamedays, and you use Fault Injection Service (FIS) to provide the controls and guardrails that teams need to run experiments in a production-like environment.

Common anti-patterns:

- There is a known performance issue in your application server. It is added to the backlog behind every planned feature implementation. If the rate of planned features being added remains constant, the performance issue would never be addressed.

- To support continual improvement, you approve administrators and developers using all their extra time to select and implement improvements. No improvements are ever completed.
- Operational acceptance is complete, and you do not test operational practices again.

Benefits of establishing this best practice: By dedicating time and resources within your processes, you can make continuous, incremental improvements possible.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Allocate time to make improvements: Dedicate time and resources within your processes to make continuous, incremental improvements.
- Implement changes to improve and evaluate the results to determine success.
- If the results do not satisfy the goals and the improvement is still a priority, pursue alternative courses of action.
- Simulate production workloads through game days, and use learnings from these simulations to improve.

Resources

Related best practices:

- OPS05-BP08 Use multiple environments

Related videos:

- AWS re:Invent 2023 - Improve application resilience with AWS Fault Injection Service

Security

The Security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security. You can find prescriptive guidance on implementation in the Security Pillar whitepaper.

Best practice areas

- Security foundations
- Identity and access management
- Detection
- Infrastructure protection
- Data protection
- Incident response
- Application security

Security foundations

Question

- SEC 1. How do you securely operate your workload?

SEC 1. How do you securely operate your workload?

To operate your workload securely, you must apply overarching best practices to every area of security. Take requirements and processes that you have defined in operational excellence at an organizational and workload level, and apply them to all areas. Staying up to date with AWS and industry recommendations and threat intelligence helps you evolve your threat model and control objectives. Automating security processes, testing, and validation permit you to scale your security operations.

Best practices

- SEC01-BP01 Separate workloads using accounts
- SEC01-BP02 Secure account root user and properties
- SEC01-BP03 Identify and validate control objectives
- SEC01-BP04 Stay up to date with security threats and recommendations
- SEC01-BP05 Reduce security management scope
- SEC01-BP06 Automate deployment of standard security controls
- SEC01-BP07 Identify threats and prioritize mitigations using a threat model
- SEC01-BP08 Evaluate and implement new security services and features regularly

SEC01-BP01 Separate workloads using accounts

Establish common guardrails and isolation between environments (such as production, development, and test) and workloads through a multi-account strategy. Account-level separation is strongly recommended, as it provides a strong isolation boundary for security, billing, and access.

Desired outcome: An account structure that isolates cloud operations, unrelated workloads, and environments into separate accounts, increasing security across the cloud infrastructure.

Common anti-patterns:

- Placing multiple unrelated workloads with different data sensitivity levels into the same account.
- Poorly defined organizational unit (OU) structure.

Benefits of establishing this best practice:

- Decreased scope of impact if a workload is inadvertently accessed.
- Central governance of access to AWS services, resources, and Regions.
- Maintain security of the cloud infrastructure with policies and centralized administration of security services.
- Automated account creation and maintenance process.
- Centralized auditing of your infrastructure for compliance and regulatory requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS accounts provide a security isolation boundary between workloads or resources that operate at different sensitivity levels. AWS provides tools to manage your cloud workloads at scale through a multi-account strategy to leverage this isolation boundary. For guidance on the concepts, patterns, and implementation of a multi-account strategy on AWS, see Organizing Your AWS Environment Using Multiple Accounts.

When you have multiple AWS accounts under central management, your accounts should be organized into a hierarchy defined by layers of organizational units (OUs). Security controls can then be organized and applied to the OUs and member accounts, establishing consistent preventative controls on member accounts in the organization. The security controls are inherited,

allowing you to filter permissions available to member accounts located at lower levels of an OU hierarchy. A good design takes advantage of this inheritance to reduce the number and complexity of security policies required to achieve the desired security controls for each member account.

AWS Organizations and AWS Control Tower are two services that you can use to implement and manage this multi-account structure in your AWS environment. AWS Organizations allows you to organize accounts into a hierarchy defined by one or more layers of OUs, with each OU containing a number of member accounts. Service control policies (SCPs) allow the organization administrator to establish granular preventative controls on member accounts, and AWS Config can be used to establish proactive and detective controls on member accounts. Many AWS services integrate with AWS Organizations to provide delegated administrative controls and performing service-specific tasks across all member accounts in the organization.

Layered on top of AWS Organizations, AWS Control Tower provides a one-click best practices setup for a multi-account AWS environment with a landing zone. The landing zone is the entry point to the multi-account environment established by Control Tower. Control Tower provides several benefits over AWS Organizations. Three benefits that provide improved account governance are:

- Integrated mandatory security controls that are automatically applied to accounts admitted into the organization.
- Optional controls that can be turned on or off for a given set of OUs.
- AWS Control Tower Account Factory provides automated deployment of accounts containing pre-approved baselines and configuration options inside your organization.

Implementation steps

1. **Design an organizational unit structure:** A properly designed organizational unit structure reduces the management burden required to create and maintain service control policies and other security controls. Your organizational unit structure should be aligned with your business needs, data sensitivity, and workload structure.
2. **Create a landing zone for your multi-account environment:** A landing zone provides a consistent security and infrastructure foundation from which your organization can quickly develop, launch, and deploy workloads. You can use a custom-built landing zone or AWS Control Tower to orchestrate your environment.
3. **Establish guardrails:** Implement consistent security guardrails for your environment through your landing zone. AWS Control Tower provides a list of mandatory and optional controls that can be deployed. Mandatory controls are automatically deployed when implementing Control

Tower. Review the list of highly recommended and optional controls, and implement controls that are appropriate to your needs.

4. **Restrict access to newly added Regions**: For new AWS Regions, IAM resources such as users and roles are only propagated to the Regions that you specify. This action can be performed through the console when using Control Tower, or by adjusting IAM permission policies in AWS Organizations.
5. **Consider AWS CloudFormation StackSets**: StackSets help you deploy resources including IAM policies, roles, and groups into different AWS accounts and Regions from an approved template.

Resources

Related best practices:

- SEC02-BP04 Rely on a centralized identity provider

Related documents:

- AWS Control Tower
- AWS Security Audit Guidelines
- IAM Best Practices
- Use CloudFormation StackSets to provision resources across multiple AWS accounts and regions
- Organizations FAQ
- AWS Organizations terminology and concepts
- Best Practices for Service Control Policies in an AWS Organizations Multi-Account Environment
- AWS Account Management Reference Guide
- Organizing Your AWS Environment Using Multiple Accounts

Related videos:

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- Building and Governing Multiple Accounts using AWS Control Tower
- Enable Control Tower for Existing Organizations

SEC01-BP02 Secure account root user and properties

The root user is the most privileged user in an AWS account, with full administrative access to all resources within the account, and in some cases cannot be constrained by security policies. Deactivating programmatic access to the root user, establishing appropriate controls for the root user, and avoiding routine use of the root user helps reduce the risk of inadvertent exposure of the root credentials and subsequent compromise of the cloud environment.

Desired outcome: Securing the root user helps reduce the chance that accidental or intentional damage can occur through the misuse of root user credentials. Establishing detective controls can also alert the appropriate personnel when actions are taken using the root user.

Common anti-patterns:

- Using the root user for tasks other than the few that require root user credentials.
- Neglecting to test contingency plans on a regular basis to verify the functioning of critical infrastructure, processes, and personnel during an emergency.
- Only considering the typical account login flow and neglecting to consider or test alternate account recovery methods.
- Not handling DNS, email servers, and telephone providers as part of the critical security perimeter, as these are used in the account recovery flow.

Benefits of establishing this best practice: Securing access to the root user builds confidence that actions in your account are controlled and audited.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS offers many tools to help secure your account. However, because some of these measures are not turned on by default, you must take direct action to implement them. Consider these recommendations as foundational steps to securing your AWS account. As you implement these steps it's important that you build a process to continuously assess and monitor the security controls.

When you first create an AWS account, you begin with one identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user. You can sign in as the root user using the email address and password that you used to create the account. Due to the elevated access granted to the AWS root user, you must limit use of the

AWS root user to perform tasks that specifically require it. The root user login credentials must be closely guarded, and multi-factor authentication (MFA) should always be used for the AWS account root user.

In addition to the normal authentication flow to log into your root user using a username, password, and multi-factor authentication (MFA) device, there are account recovery flows to log into your AWS account root user given access to the email address and phone number associated with your account. Therefore, it is equally important to secure the root user email account where the recovery email is sent and the phone number associated with the account. Also consider potential circular dependencies where the email address associated with the root user is hosted on email servers or domain name service (DNS) resources from the same AWS account.

When using AWS Organizations, there are multiple AWS accounts each of which have a root user. One account is designated as the management account and several layers of member accounts can then be added underneath the management account. Prioritize securing your management account's root user, then address your member account root users. The strategy for securing your management account's root user can differ from your member account root users, and you can place preventative security controls on your member account root users.

Implementation steps

The following implementation steps are recommended to establish controls for the root user. Where applicable, recommendations are cross-referenced to CIS AWS Foundations benchmark version 1.4.0. In addition to these steps, consult AWS best practice guidelines for securing your AWS account and resources.

Preventative controls

1. Set up accurate contact information for the account.
    a. This information is used for the lost password recovery flow, lost MFA device account recovery flow, and for critical security-related communications with your team.
    b. Use an email address hosted by your corporate domain, preferably a distribution list, as the root user's email address. Using a distribution list rather than an individual's email account provides additional redundancy and continuity for access to the root account over long periods of time.
    c. The phone number listed on the contact information should be a dedicated, secure phone for this purpose. The phone number should not be listed or shared with anyone.
2. Do not create access keys for the root user. If access keys exist, remove them (CIS 1.4).

a. Eliminate any long-lived programmatic credentials (access and secret keys) for the root user.

b. If root user access keys already exist, you should transition processes using those keys to use temporary access keys from an AWS Identity and Access Management (IAM) role, then delete the root user access keys.

3. Determine if you need to store credentials for the root user.

a. If you are using AWS Organizations to create new member accounts, the initial password for the root user on new member accounts is set to a random value that is not exposed to you. Consider using the password reset flow from your AWS Organization management account to gain access to the member account if needed.

b. For standalone AWS accounts or the management AWS Organization account, consider creating and securely storing credentials for the root user. Use MFA for the root user.

4. Use preventative controls for member account root users in AWS multi-account environments.

a. Consider using the Disallow Creation of Root Access Keys for the Root User preventative guard rail for member accounts.

b. Consider using the Disallow Actions as a Root User preventative guard rail for member accounts.

5. If you need credentials for the root user:

a. Use a complex password.

b. Turn on multi-factor authentication (MFA) for the root user, especially for AWS Organizations management (payer) accounts (CIS 1.5).

c. Consider hardware MFA devices for resiliency and security, as single use devices can reduce the chances that the devices containing your MFA codes might be reused for other purposes. Verify that hardware MFA devices powered by a battery are replaced regularly. (CIS 1.6)

- To configure MFA for the root user, follow the instructions for creating either a virtual MFA or hardware MFA device.

d. Consider enrolling multiple MFA devices for backup. Up to 8 MFA devices are allowed per account.

- Note that enrolling more than one MFA device for the root user automatically turns off the flow for recovering your account if the MFA device is lost.

e. Store the password securely, and consider circular dependencies if storing the password electronically. Don't store the password in such a way that would require access to the same AWS account to obtain it.

6. Optional: Consider establishing a periodic password rotation schedule for the root user.

- Credential management best practices depend on your regulatory and policy requirements. Root users protected by MFA are not reliant on the password as a single factor of authentication.
- Changing the root user password on a periodic basis reduces the risk that an inadvertently exposed password can be misused.

Detective controls

- Create alarms to detect use of the root credentials (CIS 1.7). Amazon GuardDuty can monitor and alert on root user API credential usage through the RootCredentialUsage finding.
- Evaluate and implement the detective controls included in the AWS Well-Architected Security Pillar conformance pack for AWS Config, or if using AWS Control Tower, the strongly recommended controls available inside Control Tower.

Operational guidance

- Determine who in the organization should have access to the root user credentials.
  - Use a two-person rule so that no one individual has access to all necessary credentials and MFA to obtain root user access.
  - Verify that the organization, and not a single individual, maintains control over the phone number and email alias associated with the account (which are used for password reset and MFA reset flow).
- Use root user only by exception (CIS 1.7).
  - The AWS root user must not be used for everyday tasks, even administrative ones. Only log in as the root user to perform AWS tasks that require root user. All other actions should be performed by other users assuming appropriate roles.
- Periodically check that access to the root user is functioning so that procedures are tested prior to an emergency situation requiring the use of the root user credentials.
- Periodically check that the email address associated with the account and those listed under Alternate Contacts work. Monitor these email inboxes for security notifications you might receive from <abuse@amazon.com>. Also ensure any phone numbers associated with the account are working.
- Prepare incident response procedures to respond to root account misuse. Refer to the AWS Security Incident Response Guide and the best practices in the Incident Response section of the

Security Pillar whitepaper for more information on building an incident response strategy for your AWS account.

Resources

Related best practices:

- SEC01-BP01 Separate workloads using accounts
- SEC02-BP01 Use strong sign-in mechanisms
- SEC03-BP02 Grant least privilege access
- SEC03-BP03 Establish emergency access process
- SEC10-BP05 Pre-provision access

Related documents:

- AWS Control Tower
- AWS Security Audit Guidelines
- IAM Best Practices
- Amazon GuardDuty – root credential usage alert
- Step-by-step guidance on monitoring for root credential use through CloudTrail
- MFA tokens approved for use with AWS
- Implementing break glass access on AWS
- Top 10 security items to improve in your AWS account
- What do I do if I notice unauthorized activity in my AWS account?

Related videos:

- Enable AWS adoption at scale with automation and governance
- Security Best Practices the Well-Architected Way
- Limiting use of AWS root credentials from AWS re:inforce 2022 – Security best practices with AWS IAM

SEC01-BP03 Identify and validate control objectives

Based on your compliance requirements and risks identified from your threat model, derive and validate the control objectives and controls that you need to apply to your workload. Ongoing validation of control objectives and controls help you measure the effectiveness of risk mitigation.

Desired outcome: The security control objectives of your business are well-defined and aligned to your compliance requirements. Controls are implemented and enforced through automation and policy and are continually evaluated for their effectiveness in achieving your objectives. Evidence of effectiveness at both a point in time and over a period of time are readily reportable to auditors.

Common anti-patterns:

- Regulatory requirements, market expectations, and industry standards for assurable security are not well-understood for your business
- Your cybersecurity frameworks and control objectives are misaligned to the requirements of your business
- The implementation of controls does not strongly align to your control objectives in a measurable way
- You do not use automation to report on the effectiveness of your controls

Level of risk exposed if this best practice is not established: High

Implementation guidance

There are many common cybersecurity frameworks that can form the basis for your security control objectives. Consider the regulatory requirements, market expectations, and industry standards for your business to determine which frameworks best supports your needs. Examples include AICPA SOC 2, HITRUST, PCI-DSS, ISO 27001, and NIST SP 800-53.

For the control objectives you identify, understand how AWS services you consume help you to achieve those objectives. Use AWS Artifact to find documentation and reports aligned to your target frameworks that describe the scope of responsibility covered by AWS and guidance for the remaining scope that is your responsibility. For further service-specific guidance as they align to various framework control statements, see AWS Customer Compliance Guides.

As you define the controls that achieve your objectives, codify enforcement using preventative controls, and automate mitigations using detective controls. Help prevent non-compliant resource

configurations and actions across your AWS Organizations using service control policies (SCP). Implement rules in AWS Config to monitor and report on non-compliant resources, then switch rules to an enforcement model once confident in their behavior. To deploy sets of pre-defined and managed rules that align to your cybersecurity frameworks, evaluate the use of AWS Security Hub CSPM standards as your first option. The AWS Foundational Service Best Practices (FSBP) standard and the CIS AWS Foundations Benchmark are good starting points with controls that align to many objectives that are shared across multiple standard frameworks. Where Security Hub CSPM does not intrinsically have the control detections desired, it can be complemented using AWS Config conformance packs.

Use APN Partner Bundles recommended by the AWS Global Security and Compliance Acceleration (GSCA) team to get assistance from security advisors, consulting agencies, evidence collection and reporting systems, auditors, and other complementary services when required.

Implementation steps

1. Evaluate common cybersecurity frameworks, and align your control objectives to the ones chosen.
2. Obtain relevant documentation on guidance and responsibilities for your framework using AWS Artifact. Understand which parts of compliance fall on the AWS side of the shared responsibility model and which parts are your responsibility.
3. Use SCPs, resource policies, role trust policies, and other guardrails to prevent non-compliant resource configurations and actions.
4. Evaluate deploying Security Hub CSPM standards and AWS Config conformance packs that align to your control objectives.

Resources

Related best practices:

- SEC03-BP01 Define access requirements
- SEC04-BP01 Configure service and application logging
- SEC07-BP01 Understand your data classification scheme
- OPS01-BP03 Evaluate governance requirements
- OPS01-BP04 Evaluate compliance requirements
- PERF01-BP05 Use policies and reference architectures

- COST02-BP01 Develop policies based on your organization requirements

Related documents:

- AWS Customer Compliance Guides

Related tools:

- AWS Artifact

SEC01-BP04 Stay up to date with security threats and recommendations

Stay up to date with the latest threats and mitigations by monitoring industry threat intelligence publications and data feeds for updates. Evaluate managed service offerings that automatically update based on the latest threat data.

Desired outcome: You stay informed as industry publications are updated with the latest threats and recommendations. You use automation to detect potential vulnerabilities and exposures as you identify new threats. You take mitigating action against these threats. You adopt AWS services that automatically update with the latest threat intelligence.

Common anti-patterns:

- Not having a reliable and repeatable mechanism to stay informed of the latest threat intelligence.
- Maintaining manual inventory of your technology portfolio, workloads, and dependencies that require human review for potential vulnerabilities and exposures.
- Not having mechanisms in place to update your workloads and dependencies to the latest versions available that provide known threat mitigations.

Benefits of establishing this best practice: Using threat intelligence sources to stay up to date reduces the risk of missing out on important changes to the threat landscape that can impact your business. Having automation in place to scan, detect, and remediate where potential vulnerabilities or exposures exist in your workloads and their dependencies can help you mitigate risks quickly and predictably, compared to manual alternatives. This helps control time and costs related to vulnerability mitigation.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Review trusted threat intelligence publications to stay on top of the threat landscape. Consult the MITRE ATT&CK knowledge base for documentation on known adversarial tactics, techniques, and procedures (TTPs). Review MITRE's Common Vulnerabilities and Exposures (CVE) list to stay informed on known vulnerabilities in products you rely on. Understand critical risks to web applications with the Open Worldwide Application Security Project (OWASP)'s popular OWASP Top 10 project.

Stay up to date on AWS security events and recommended remediation steps with AWS Security Bulletins for CVEs.

To reduce your overall effort and overhead of staying up to date, consider using AWS services that automatically incorporate new threat intelligence over time. For example, Amazon GuardDuty stays up to date with industry threat intelligence for detecting anomalous behaviors and threat signatures within your accounts. Amazon Inspector automatically keeps a database of the CVEs it uses for its continuous scanning features up to date. Both AWS WAF and AWS Shield Advanced provide managed rule groups that are updated automatically as new threats emerge.

Review the Well-Architected operational excellence pillar for automated fleet management and patching.

Implementation steps

- Subscribe to updates for threat intelligence publications that are relevant to your business and industry. Subscribe to the AWS Security Bulletins.
- Consider adopting services that incorporate new threat intelligence automatically, such as Amazon GuardDuty and Amazon Inspector.
- Deploy a fleet management and patching strategy that aligns with the best practices of the Well-Architected Operational Excellence Pillar.

Resources

Related best practices:

- SEC01-BP07 Identify threats and prioritize mitigations using a threat model
- OPS01-BP05 Evaluate threat landscape
- OPS11-BP01 Have a process for continuous improvement

SEC01-BP05 Reduce security management scope

Determine if you can reduce your security scope by using AWS services that shift management of certain controls to AWS (*managed services*). These services can help reduce your security maintenance tasks, such as infrastructure provisioning, software setup, patching, or backups.

Desired outcome: You consider the scope of your security management when selecting AWS services for your workload. The cost of management overhead and maintenance tasks (the total cost of ownership, or TCO) is weighed against the cost of the services you select, in addition to other Well-Architected considerations. You incorporate AWS control and compliance documentation into your control evaluation and verification procedures.

Common anti-patterns:

- Deploying workloads without thoroughly understanding the shared responsibility model for the services you select.
- Hosting databases and other technologies on virtual machines without having evaluated a managed service equivalent.
- Not including security management tasks into the total cost of ownership of hosting technologies on virtual machines when compared to managed service options.

Benefits of establishing this best practice: Using managed services can reduce your overall burden of managing operational security controls, which can reduce your security risks and total cost of ownership. Time that would otherwise be spent on certain security tasks can be reinvested into tasks that provide more value to your business. Managed services can also reduce the scope of your compliance requirements by shifting some control requirements to AWS.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

There are multiple ways you can integrate the components of your workload on AWS. Installing and running technologies on Amazon EC2 instances often requires you to take on the largest share of the overall security responsibility. To help reduce the burden of operating certain controls, identify AWS managed services that reduce the scope of your side of the shared responsibility model and understand how you can use them in your existing architecture. Examples include using the Amazon Relational Database Service (Amazon RDS) for deploying databases, Amazon Elastic Kubernetes Service (Amazon EKS) or Amazon Elastic Container Service (Amazon ECS)

for orchestrating containers, or using serverless options. When building new applications, think through which services can help reduce time and cost when it comes to implementing and managing security controls.

Compliance requirements can also be a factor when selecting services. Managed services can shift the compliance of some requirements to AWS. Discuss with your compliance team about their degree of comfort with auditing the aspects of services you operate and manage and accepting control statements in relevant AWS audit reports. You can provide the audit artifacts found in AWS Artifact to your auditors or regulators as evidence of AWS security controls. You can also use the responsibility guidance provided by some of the AWS audit artifacts to design your architecture, along with the AWS Customer Compliance Guides. This guidance helps determine the additional security controls you should put in place in order to support the specific use cases of your system.

When using managed services, be familiar with the process of updating their resources to newer versions (for example, updating the version of a database managed by Amazon RDS, or a programming language runtime for an AWS Lambda function). While the managed service may perform this operation for you, configuring the timing of the update and understanding the impact on your operations remains your responsibility. Tools like AWS Health can help you track and manage these updates throughout your environments.

Implementation steps

1. Evaluate the components of your workload that can be replaced with a managed service.
    a. If you are migrating a workload to AWS, consider the reduced management (time and expense) and reduction of risk when you assess if you should rehost, refactor, replatform, rebuild, or replace your workload. Sometimes additional investment at the start of a migration can have significant savings in the long run.
2. Consider implementing managed services, like Amazon RDS, instead of installing and managing your own technology deployments.
3. Use the responsibility guidance in AWS Artifact to help determine the security controls you should put in place for your workload.
4. Keep an inventory of resources in use, and stay up-to-date with new services and approaches to identify new opportunities to reduce scope.

Resources

Related best practices:

- PERF02-BP01 Select the best compute options for your workload
- PERF03-BP01 Use a purpose-built data store that best supports your data access and storage requirements
- SUS05-BP03 Use managed services

Related documents:

- Planned lifecycle events for AWS Health

Related tools:

- AWS Health
- AWS Artifact
- AWS Customer Compliance Guides

Related videos:

- How do I migrate to an Amazon RDS or Aurora MySQL DB instance using AWS DMS?
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health

SEC01-BP06 Automate deployment of standard security controls

Apply modern DevOps practices as you develop and deploy security controls that are standard across your AWS environments. Define standard security controls and configurations using Infrastructure as Code (IaC) templates, capture changes in a version control system, test changes as part of a CI/CD pipeline, and automate the deployment of changes to your AWS environments.

Desired outcome: IaC templates capture standardized security controls and commit them to a version control system. CI/CD pipelines are in places that detect changes and automate testing and deploying your AWS environments. Guardrails are in place to detect and alert on misconfigurations in templates before proceeding to deployment. Workloads are deployed into environments where standard controls are in place. Teams have access to deploy approved service configurations through a self-service mechanism. Secure backup and recovery strategies are in place for control configurations, scripts, and related data.

Common anti-patterns:

- Making changes to your standard security controls manually, through a web console or command-line interface.
- Relying on individual workload teams to manually implement the controls a central team defines.
- Relying on a central security team to deploy workload-level controls at the request of a workload team.
- Allowing the same individuals or teams to develop, test, and deploy security control automation scripts without proper separation of duties or checks and balances.

Benefits of establishing this best practice: Using templates to define your standard security controls allows you to track and compare changes over time using a version control system. Using automation to test and deploy changes creates standardization and predictability, increasing the chances of a successful deployment and reducing manual repetitive tasks. Providing a self-serve mechanism for workload teams to deploy approved services and configurations reduces the risk of misconfiguration and misuse. This also helps them to incorporate controls earlier in the development process.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

When following the practices described in SEC01-BP01 Separate workloads using accounts, you will end up with multiple AWS accounts for different environments that you manage using AWS Organizations. While each of these environments and workloads may need distinct security controls, you can standardize some security controls across your organization. Examples include integrating centralized identity providers, defining networks and firewalls, and configuring standard locations for storing and analyzing logs. In the same way you can use *infrastructure as code* (IaC) to apply the same rigor of application code development to infrastructure provisioning, you can use IaC to define and deploy your standard security controls as well.

Wherever possible, define your security controls in a declarative way, such as in AWS CloudFormation, and store them in a source control system. Use DevOps practices to automate the deploying your controls for more predictable releases, automated testing using tools like AWS CloudFormation Guard, and detecting drift between your deployed controls and your desired configuration. You can use services such as AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy to construct a CI/CD pipeline. Consider the guidance in Organizing Your AWS

Environment Using Multiple Accounts to configure these services in their own accounts that are separate from other deployment pipelines.

You can also define templates to standardize defining and deploying AWS accounts, services, and configurations. This technique allows a central security team to manage these definitions and provide them to workload teams through a self-service approach. One way to achieve this is by using Service Catalog, where you can publish templates as *products* that workload teams can incorporate into their own pipeline deployments. If you are using AWS Control Tower, some templates and controls are available as a starting point. Control Tower also provides the Account Factory capability, allowing workload teams to create new AWS accounts using the standards you define. This capability helps remove dependencies on a central team to approve and create new accounts when they are identified as needed by your workload teams. You may need these accounts to isolate different workload components based on reasons such as the function they serve, the sensitivity of data being processed, or their behavior.

Implementation steps

1. Determine how you will store and maintain your templates in a version control system.
2. Create CI/CD pipelines to test and deploy your templates. Define tests to check for misconfigurations and that templates adhere to your company standards.
3. Build a catalog of standardized templates for workload teams to deploy AWS accounts and services according to your requirements.
4. Implement secure backup and recovery strategies for your control configurations, scripts, and related data.

Resources

Related best practices:

- OPS05-BP01 Use version control
- OPS05-BP04 Use build and deployment management systems
- REL08-BP05 Deploy changes with automation
- SUS06-BP01 Adopt methods that can rapidly introduce sustainability improvements

Related documents:

- Organizing Your AWS Environment Using Multiple Accounts

Related examples:

- Automate account creation, and resource provisioning using Service Catalog, AWS Organizations, and AWS Lambda
- Strengthen the DevOps pipeline and protect data with AWS Secrets Manager, AWS KMS, and AWS Certificate Manager

Related tools:

- AWS CloudFormation Guard
- Landing Zone Accelerator on AWS

SEC01-BP07 Identify threats and prioritize mitigations using a threat model

Perform threat modeling to identify and maintain an up-to-date register of potential threats and associated mitigations for your workload. Prioritize your threats and adapt your security control mitigations to prevent, detect, and respond. Revisit and maintain this in the context of your workload, and the evolving security landscape.

Level of risk exposed if this best practice is not established: High

Implementation guidance

What is threat modeling?

“Threat modeling works to identify, communicate, and understand threats and mitigations within the context of protecting something of value.” – The Open Web Application Security Project (OWASP) Application Threat Modeling

Why should you threat model?

Systems are complex, and are becoming increasingly more complex and capable over time, delivering more business value and increased customer satisfaction and engagement. This means that IT design decisions need to account for an ever-increasing number of use cases. This complexity and number of use-case permutations typically makes unstructured approaches ineffective for finding and mitigating threats. Instead, you need a systematic approach to enumerate the potential threats to the system, and to devise mitigations and prioritize these mitigations to make sure that the limited resources of your organization have the maximum impact in improving the overall security posture of the system.

Threat modeling is designed to provide this systematic approach, with the aim of finding and addressing issues early in the design process, when the mitigations have a low relative cost and effort compared to later in the lifecycle. This approach aligns with the industry principle of *shift-left* security. Ultimately, threat modeling integrates with an organization's risk management process and helps drive decisions on which controls to implement by using a threat driven approach.

When should threat modeling be performed?

Start threat modeling as early as possible in the lifecycle of your workload, this gives you better flexibility on what to do with the threats you have identified. Much like software bugs, the earlier you identify threats, the more cost effective it is to address them. A threat model is a living document and should continue to evolve as your workloads change. Revisit your threat models over time, including when there is a major change, a change in the threat landscape, or when you adopt a new feature or service.

Implementation steps

How can we perform threat modeling?

There are many different ways to perform threat modeling. Much like programming languages, there are advantages and disadvantages to each, and you should choose the way that works best for you. One approach is to start with Shostack's 4 Question Frame for Threat Modeling, which poses open-ended questions to provide structure to your threat modeling exercise:

1. **What are we working on?**

   The purpose of this question is to help you understand and agree upon the system you are building and the details about that system that are relevant to security. Creating a model or diagram is the most popular way to answer this question, as it helps you to visualize what you are building, for example, using a data flow diagram. Writing down assumptions and important details about your system also helps you define what is in scope. This allows everyone contributing to the threat model to focus on the same thing, and avoid time-consuming detours into out-of-scope topics (including out of date versions of your system). For example, if you are building a web application, it is probably not worth your time threat modeling the operating system trusted boot sequence for browser clients, as you have no ability to affect this with your design.

2. **What can go wrong?**

This is where you identify threats to your system. Threats are accidental or intentional actions or events that have unwanted impacts and could affect the security of your system. Without a clear understanding of what could go wrong, you have no way of doing anything about it.

There is no canonical list of what can go wrong. Creating this list requires brainstorming and collaboration between all of the individuals within your team and relevant personas involved in the threat modeling exercise. You can aid your brainstorming by using a model for identifying threats, such as STRIDE, which suggests different categories to evaluate: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of privilege. In addition, you might want to aid the brainstorming by reviewing existing lists and research for inspiration, including the OWASP Top 10, HiTrust Threat Catalog, and your organization's own threat catalog.

3. **What are we going to do about it?**

   As was the case with the previous question, there is no canonical list of all possible mitigations. The inputs into this step are the identified threats, actors, and areas of improvement from the previous step.

   Security and compliance is a shared responsibility between you and AWS. It's important to understand that when you ask "What are we going to do about it?", that you are also asking "Who is responsible for doing something about it?". Understanding the balance of responsibilities between you and AWS helps you scope your threat modeling exercise to the mitigations that are under your control, which are typically a combination of AWS service configuration options and your own system-specific mitigations.

   For the AWS portion of the shared responsibility, you will find that AWS services are in-scope of many compliance programs. These programs help you to understand the robust controls in place at AWS to maintain security and compliance of the cloud. The audit reports from these programs are available for download for AWS customers from AWS Artifact.

   Regardless of which AWS services you are using, there's always an element of customer responsibility, and mitigations aligned to these responsibilities should be included in your threat model. For security control mitigations for the AWS services themselves, you want to consider implementing security controls across domains, including domains such as identity and access management (authentication and authorization), data protection (at rest and in transit), infrastructure security, logging, and monitoring. The documentation for each AWS service has a dedicated security chapter that provides guidance on the security controls to consider as

mitigations. Importantly, consider the code that you are writing and its code dependencies, and think about the controls that you could put in place to address those threats. These controls could be things such as input validation, session handling, and bounds handling. Often, the majority of vulnerabilities are introduced in custom code, so focus on this area.

4. **Did we do a good job?**

   The aim is for your team and organization to improve both the quality of threat models and the velocity at which you are performing threat modeling over time. These improvements come from a combination of practice, learning, teaching, and reviewing. To go deeper and get hands on, it's recommended that you and your team complete the Threat modeling the right way for builders training course or workshop. In addition, if you are looking for guidance on how to integrate threat modeling into your organization's application development lifecycle, see How to approach threat modeling post on the AWS Security Blog.

Threat Composer

To aid and guide you in performing threat modeling, consider using the Threat Composer tool, which aims to your reduce time-to-value when threat modeling. The tool helps you do the following:

- Write useful threat statements aligned to threat grammar that work in a natural non-linear workflow
- Generate a human-readable threat model
- Generate a machine-readable threat model to allow you treat threat models as code
- Help you to quickly identify areas of quality and coverage improvement using the Insights Dashboard

For further reference, visit Threat Composer and switch to the system-defined **Example Workspace**.

Resources

Related best practices:

- SEC01-BP03 Identify and validate control objectives
- SEC01-BP04 Stay up to date with security threats and recommendations
- SEC01-BP05 Reduce security management scope

- SEC01-BP08 Evaluate and implement new security services and features regularly

Related documents:

- How to approach threat modeling (AWS Security Blog)
- NIST: Guide to Data-Centric System Threat modeling

Related videos:

- AWS Summit ANZ 2021 - How to approach threat modeling
- AWS Summit ANZ 2022 - Scaling security – Optimise for fast and secure delivery

Related training:

- Threat modeling the right way for builders – AWS Skill Builder virtual self-paced training
- Threat modeling the right way for builders – AWS Workshop

Related tools:

- Threat Composer

SEC01-BP08 Evaluate and implement new security services and features regularly

Evaluate and implement security services and features from AWS and AWS Partners that help you evolve the security posture of your workload.

Desired outcome: You have a standard practice in place that informs you of new features and services released by AWS and AWS Partners. You evaluate how these new capabilities influence the design of current and new controls for your environments and workloads.

Common anti-patterns:

- You don't subscribe to AWS blogs and RSS feeds to learn of relevant new features and services quickly
- You rely on news and updates about security services and features from second-hand sources
- You don't encourage AWS users in your organization to stay informed on the latest updates

Benefits of establishing this best practice: When you stay on top of new security services and features, you can make informed decisions about the implementation of controls in your cloud environments and workloads. These sources help raise awareness of the evolving security landscape and how AWS services can be used to protect against new and emerging threats.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

AWS informs customers of new security services and features through several channels:

- AWS What's New
- AWS News Blog
- AWS Security Blog
- AWS Security Bulletins
- AWS documentation overview

You can subscribe to an AWS Daily Feature Updates topic using Amazon Simple Notification Service (Amazon SNS) for a comprehensive daily summary of updates. Some security services, such as Amazon GuardDuty and AWS Security Hub CSPM, provide their own SNS topics to stay informed about new standards, findings, and other updates for those particular services.

New services and features are also announced and described in detail during conferences, events, and webinars conducted around the globe each year. Of particular note is the annual AWS re:Inforce security conference and the more general AWS re:Invent conference. The previously-mentioned AWS news channels share these conference announcements about security and other services, and you can view deep dive educational breakout sessions online at the AWS Events channel on YouTube.

You can also ask your AWS account team about the latest security service updates and recommendations. You can reach out to your team through the Sales Support form if you do not have their direct contact information. Similarly, if you subscribed to AWS Enterprise Support, you will receive weekly updates from your Technical Account Manager (TAM) and can schedule a regular review meeting with them.

Implementation steps

1. Subscribe to the various blogs and bulletins with your favorite RSS reader or to the Daily Features Updates SNS topic.

2. Evaluate which AWS events to attend to learn first-hand about new features and services.
3. Set up meetings with your AWS account team for any questions about updating security services and features.
4. Consider subscribing to Enterprise Support to have regular consultations with a Technical Account Manager (TAM).

Resources

Related best practices:

- PERF01-BP01 Learn about and understand available cloud services and features
- COST01-BP07 Keep up-to-date with new service releases

Identity and access management

Questions

- SEC 2. How do you manage authentication for people and machines?
- SEC 3. How do you manage permissions for people and machines?

SEC 2. How do you manage authentication for people and machines?

There are two types of identities you need to manage when approaching operating secure AWS workloads.

- **Human identities:** The human identities that require access to your AWS environments and applications can be categorized into three groups: workforce, third parties, and users.

  The *workforce* group includes administrators, developers, and operators who are members of your organization. They need access to manage, build, and operate your AWS resources.

  Third parties are external collaborators, such as contractors, vendors, or partners. They interact with your AWS resources as part of their engagement with your organization.

  Users are the consumers of your applications. They access your AWS resources through web browsers, client applications, mobile apps, or interactive command-line tools.
- **Machine identities:** Your workload applications, operational tools, and components require an identity to make requests to AWS services, such as reading data. These identities also include

machines running within your AWS environment, like Amazon EC2 instances or AWS Lambda functions. You may also manage machine identities for external parties, or machines outside of AWS, that require access to your AWS environment.

Best practices

- SEC02-BP01 Use strong sign-in mechanisms
- SEC02-BP02 Use temporary credentials
- SEC02-BP03 Store and use secrets securely
- SEC02-BP04 Rely on a centralized identity provider
- SEC02-BP05 Audit and rotate credentials periodically
- SEC02-BP06 Employ user groups and attributes

SEC02-BP01 Use strong sign-in mechanisms

Sign-ins (authentication using sign-in credentials) can present risks when not using mechanisms like multi-factor authentication (MFA), especially in situations where sign-in credentials have been inadvertently disclosed or are easily guessed. Use strong sign-in mechanisms to reduce these risks by requiring MFA and strong password policies.

Desired outcome: Reduce the risks of unintended access to credentials in AWS by using strong sign-in mechanisms for AWS Identity and Access Management (IAM) users, the AWS account root user, AWS IAM Identity Center, and third-party identity providers. This means requiring MFA, enforcing strong password policies, and detecting anomalous login behavior.

Common anti-patterns:

- Not enforcing a strong password policy for your identities including complex passwords and MFA.
- Sharing the same credentials among different users.
- Not using detective controls for suspicious sign-ins.

Level of risk exposed if this best practice is not established: High

Implementation guidance

There are several ways for human identities to sign in to AWS. It is an AWS best practice to rely on a centralized identity provider using federation (direct SAML 2.0 federation between AWS IAM and

the centralized IdP or using AWS IAM Identity Center) when authenticating to AWS. In this case, establish a secure sign-in process with your identity provider or Microsoft Active Directory.

When you first open an AWS account, you begin with an AWS account root user. You should only use the account root user to set up access for your users (and for tasks that require the root user). It's important to turn on multi-factor authentication (MFA) for the account root user immediately after opening your AWS account and to secure the root user using the AWS best practice guide.

AWS IAM Identity Center is designed for workforce users, and you can create and manage user identities within the service and secure the sign-in process with MFA. AWS Cognito, on the other hand, is designed for customer identity and access management (CIAM), which provides user pools and identity providers for external user identities in your applications.

If you create users in AWS IAM Identity Center, secure the sign-in process in that service and turn on MFA. For external user identities in your applications, you can use Amazon Cognito user pools and secure the sign-in process in that service or through one of the supported identity providers in Amazon Cognito user pools.

Additionally, for users in AWS IAM Identity Center, you can use AWS Verified Access to provide an additional layer of security by verifying the user's identity and device posture before they are granted access to AWS resources.

If you are using AWS Identity and Access Management (IAM) users, secure the sign-in process using IAM.

You can use both AWS IAM Identity Center and direct IAM federation simultaneously to manage access to AWS. You can use IAM federation to manage access to the AWS Management Console and services and IAM Identity Center to manage access to business applications like Quick or Amazon Q Business.

Regardless of the sign-in method, it's critical to enforce a strong sign-in policy.

Implementation steps

The following are general strong sign-in recommendations. The actual settings you configure should be set by your company policy or use a standard like NIST 800-63.

- Require MFA. It's an IAM best practice to require MFA for human identities and workloads. Turning on MFA provides an additional layer of security requiring that users provide sign-in credentials and a one-time password (OTP) or a cryptographically verified and generated string from a hardware device.

- Enforce a minimum password length, which is a primary factor in password strength.
- Enforce password complexity to make passwords more difficult to guess.
- Allow users to change their own passwords.
- Create individual identities instead of shared credentials. By creating individual identities, you can give each user a unique set of security credentials. Individual users provide the ability to audit each user's activity.

IAM Identity Center recommendations:

- IAM Identity Center provides a predefined password policy when using the default directory that establishes password length, complexity, and reuse requirements.
- Turn on MFA and configure the context-aware or always-on setting for MFA when the identity source is the default directory, AWS Managed Microsoft AD, or AD Connector.
- Allow users to register their own MFA devices.

Amazon Cognito user pools directory recommendations:

- Configure the Password strength settings.
- Require MFA for users.
- Use the Amazon Cognito user pools advanced security settings for features like adaptive authentication which can block suspicious sign-ins.

IAM user recommendations:

- Ideally you are using IAM Identity Center or direct federation. However, you might have the need for IAM users. In that case, set a password policy for IAM users. You can use the password policy to define requirements such as minimum length or whether the password requires non-alphabetic characters.
- Create an IAM policy to enforce MFA sign-in so that users are allowed to manage their own passwords and MFA devices.

Resources

Related best practices:

- SEC02-BP03 Store and use secrets securely
- SEC02-BP04 Rely on a centralized identity provider
- SEC03-BP08 Share resources securely within your organization

Related documents:

- AWS IAM Identity Center Password Policy
- IAM user password policy
- Setting the AWS account root user password
- Amazon Cognito password policy
- AWS credentials
- IAM security best practices

Related videos:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP02 Use temporary credentials

When doing any type of authentication, it's best to use temporary credentials instead of long-term credentials to reduce or eliminate risks, such as credentials being inadvertently disclosed, shared, or stolen.

Desired outcome: To reduce the risk of long-term credentials, use temporary credentials wherever possible for both human and machine identities. Long-term credentials create many risks, such as exposure through uploads to public repositories. By using temporary credentials, you significantly reduce the chances of credentials becoming compromised.

Common anti-patterns:

- Developers using long-term access keys from IAM users rather than obtaining temporary credentials from the CLI using federation.
- Developers embedding long-term access keys in their code and uploading that code to public Git repositories.

- Developers embedding long-term access keys in mobile apps that are then made available in app stores.
- Users sharing long-term access keys with other users, or employees leaving the company with long-term access keys still in their possession.
- Using long-term access keys for machine identities when temporary credentials could be used.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Use temporary security credentials instead of long-term credentials for all AWS API and CLI requests. API and CLI requests to AWS services must, in nearly every case, be signed using AWS access keys. These requests can be signed with either temporary or long-term credentials. The only time you should use long-term credentials, also known as long-term access keys, is if you are using an IAM user or the AWS account root user. When you federate to AWS or assume an IAM role through other methods, temporary credentials are generated. Even when you access the AWS Management Console using sign-in credentials, temporary credentials are generated for you to make calls to AWS services. There are few situations where you need long-term credentials and you can accomplish nearly all tasks using temporary credentials.

Avoiding the use of long-term credentials in favor of temporary credentials should go hand in hand with a strategy of reducing the usage of IAM users in favor of federation and IAM roles. While IAM users have been used for both human and machine identities in the past, we now recommend not using them to avoid the risks in using long-term access keys.

Implementation steps

Human identities

For workforce identities like employees, administrators, developers, and operators:

- You should rely on a centralized identity provider and require human users to use federation with an identity provider to access AWS using temporary credentials. Federation for your users can be done either with direct federation to each AWS account or using AWS IAM Identity Center and the identity provider of your choice. Federation provides a number of advantages over using IAM users in addition to eliminating long-term credentials. Your users can also request temporary credentials from the command line for direct federation or by using IAM Identity Center. This means that there are few uses cases that require IAM users or long-term credentials for your users.

For third-party identities:

- When granting third parties, such as software as a service (SaaS) providers, access to resources in your AWS account, you can use cross-account roles and resource-based policies. Additionally, you can use the Amazon Cognito OAuth 2.0 grant client credentials flow for B2B SaaS customers or partners.

User identities that access your AWS resources through web browsers, client applications, mobile apps, or interactive command-line tools:

- If you need to grant applications for consumers or customers access to your AWS resources, you can use Amazon Cognito identity pools or Amazon Cognito user pools to provide temporary credentials. The permissions for the credentials are configured through IAM roles. You can also define a separate IAM role with limited permissions for guest users who are not authenticated.

Machine identities

For machine identities, you might need to use long-term credentials. In these cases, you should require workloads to use temporary credentials with IAM roles to access AWS.

- For Amazon Elastic Compute Cloud (Amazon EC2), you can use roles for Amazon EC2.
- AWS Lambda allows you to configure a Lambda execution role to grant the service permissions to perform AWS actions using temporary credentials. There are many other similar models for AWS services to grant temporary credentials using IAM roles.
- For IoT devices, you can use the AWS IoT Core credential provider to request temporary credentials.
- For on-premises systems or systems that run outside of AWS that need access to AWS resources, you can use IAM Roles Anywhere.

There are scenarios where temporary credentials are not supported, which require the use of long-term credentials. In these situations, audit and rotate these credentials periodically and rotate access keys regularly. For highly restricted IAM user access keys, consider the following additional security measures:

- Grant highly restricted permissions:
  - Adhere to the principle of least privilege (be specific about actions, resources, and conditions).

  - Consider granting the IAM user only the AssumeRole operation for one specific role. Depending on the on-premise architecture, this approach helps isolate and secure the long-term IAM credentials.
- Limit the allowed network sources and IP addresses in the IAM role trust policy.
- Monitor usage and set up alerts for unused permissions or misuse (using AWS CloudWatch Logs metric filters and alarms).
- Enforce permission boundaries (service control policies (SCPs) and permission boundaries complement each other - SCPs are coarse-grained, while permission boundaries are fine-grained).
- Implement a process to provision and securely store (in an on-premise vault) the credentials.

Some other options for scenarios requiring long-term credentials include:

- Build your own token vending API (using Amazon API Gateway).
- For scenarios where you must use long-term credentials or credentials other than AWS access keys (such as database logins), you can use a service designed to handle the management of secrets, such as AWS Secrets Manager. Secrets Manager simplifies the management, rotation, and secure storage of encrypted secrets. Many AWS services support a direct integration with Secrets Manager.
- For multi-cloud integrations, you can use identity federation based on your source credential service provider (CSP) credentials (see AWS STS AssumeRoleWithWebIdentity).

For more information about rotating long-term credentials, see rotating access keys.

Resources

Related best practices:

- SEC02-BP03 Store and use secrets securely
- SEC02-BP04 Rely on a centralized identity provider
- SEC03-BP08 Share resources securely within your organization

Related documents:

- Temporary Security Credentials

- AWS Credentials
- IAM Security Best Practices
- IAM Roles
- IAM Identity Center
- Identity Providers and Federation
- Rotating Access Keys
- Security Partner Solutions: Access and Access Control
- The AWS Account Root User
- Access AWS using a Google Cloud Platform native workload identity
- How to access AWS resources from Microsoft Entra ID tenants using AWS Security Token Service

Related videos:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP03 Store and use secrets securely

A workload requires an automated capability to prove its identity to databases, resources, and third-party services. This is accomplished using secret access credentials, such as API access keys, passwords, and OAuth tokens. Using a purpose-built service to store, manage, and rotate these credentials helps reduce the likelihood that those credentials become compromised.

Desired outcome: Implementing a mechanism for securely managing application credentials that achieves the following goals:

- Identifying what secrets are required for the workload.
- Reducing the number of long-term credentials required by replacing them with short-term credentials when possible.
- Establishing secure storage and automated rotation of remaining long-term credentials.
- Auditing access to secrets that exist in the workload.
- Continual monitoring to verify that no secrets are embedded in source code during the development process.

- Reduce the likelihood of credentials being inadvertently disclosed.

Common anti-patterns:

- Not rotating credentials.
- Storing long-term credentials in source code or configuration files.
- Storing credentials at rest unencrypted.

Benefits of establishing this best practice:

- Secrets are stored encrypted at rest and in transit.
- Access to credentials is gated through an API (think of it as a *credential vending machine*).
- Access to a credential (both read and write) is audited and logged.
- Separation of concerns: credential rotation is performed by a separate component, which can be segregated from the rest of the architecture.
- Secrets are automatically distributed on-demand to software components and rotation occurs in a central location.
- Access to credentials can be controlled in a fine-grained manner.

Level of risk exposed if this best practice is not established: High

Implementation guidance

In the past, credentials used to authenticate to databases, third-party APIs, tokens, and other secrets might have been embedded in source code or in environment files. AWS provides several mechanisms to store these credentials securely, automatically rotate them, and audit their usage.

The best way to approach secrets management is to follow the guidance of remove, replace, and rotate. The most secure credential is one that you do not have to store, manage, or handle. There might be credentials that are no longer necessary to the functioning of the workload that can be safely removed.

For credentials that are still required for the proper functioning of the workload, there might be an opportunity to replace a long-term credential with a temporary or short-term credential. For example, instead of hard-coding an AWS secret access key, consider replacing that long-term credential with a temporary credential using IAM roles.

Some long-lived secrets might not be able to be removed or replaced. These secrets can be stored in a service such as AWS Secrets Manager, where they can be centrally stored, managed, and rotated on a regular basis.

An audit of the workload's source code and configuration files can reveal many types of credentials. The following table summarizes strategies for handling common types of credentials:

Credential type	Description	Suggested strategy
IAM access keys	AWS IAM access and secret keys used to assume IAM roles inside of a workload	Replace: Use IAM roles assigned to the compute instances (such as Amazon EC2 or AWS Lambda) instead. For interoperability with third parties that require access to resources in your AWS account, ask if they support AWS cross-account access. For mobile apps, consider using temporary credentia ls through Amazon Cognito identity pools (federated identities). For workloads running outside of AWS, consider IAM Roles Anywhere or AWS Systems Manager Hybrid Activations. For containers see Amazon ECS task IAM role or Amazon EKS node IAM role.
SSH keys	Secure Shell private keys used to log into Linux EC2 instances, manually or as part of an automated process	Replace: Use AWS Systems Manager or EC2 Instance Connect to provide programmatic and human access to EC2 instances using IAM roles.

Credential type	Description	Suggested strategy
Application and database credentials	Passwords – plain text string	Rotate: Store credentials in AWS Secrets Manager and establish automated rotation if possible.
Amazon RDS and Aurora Admin Database credentials	Passwords – plain text string	Replace: Use the Secrets Manager integration with Amazon RDS or Amazon Aurora. In addition, some RDS database types can use IAM roles instead of passwords for some use cases (for more detail, see IAM database authentication).
OAuth tokens	Secret tokens – plain text string	Rotate: Store tokens in AWS Secrets Manager and configure automated rotation.
API tokens and keys	Secret tokens – plain text string	Rotate: Store in AWS Secrets Manager and establish automated rotation if possible.

A common anti-pattern is embedding IAM access keys inside source code, configuration files, or mobile apps. When an IAM access key is required to communicate with an AWS service, use temporary (short-term) security credentials. These short-term credentials can be provided through IAM roles for EC2 instances, execution roles for Lambda functions, Cognito IAM roles for mobile user access, and IoT Core policies for IoT devices. When interfacing with third parties, prefer delegating access to an IAM role with the necessary access to your account's resources rather than configuring an IAM user and sending the third party the secret access key for that user.

There are many cases where the workload requires the storage of secrets necessary to interoperate with other services and resources. AWS Secrets Manager is purpose built to securely manage

these credentials, as well as the storage, use, and rotation of API tokens, passwords, and other credentials.

AWS Secrets Manager provides five key capabilities to ensure the secure storage and handling of sensitive credentials: encryption at rest, encryption in transit, comprehensive auditing, fine-grained access control, and extensible credential rotation. Other secret management services from AWS Partners or locally developed solutions that provide similar capabilities and assurances are also acceptable.

When you retrieve a secret, you can use the Secrets Manager client side caching components to cache it for future use. Retrieving a cached secret is faster than retrieving it from Secrets Manager. Additionally, because there is a cost for calling Secrets Manager APIs, using a cache can reduce your costs. For all of the ways you can retrieve secrets, see Get secrets.

> **Note**
>
> Some languages may require you to implement your own in-memory encryption for client side caching.

Implementation steps

1. Identify code paths containing hard-coded credentials using automated tools such as Amazon CodeGuru.
   a. Use Amazon CodeGuru to scan your code repositories. Once the review is complete, filter on Type=Secrets in CodeGuru to find problematic lines of code.
2. Identify credentials that can be removed or replaced.
   a. Identify credentials no longer needed and mark for removal.
   b. For AWS Secret Keys that are embedded in source code, replace them with IAM roles associated with the necessary resources. If part of your workload is outside AWS but requires IAM credentials to access AWS resources, consider IAM Roles Anywhere or AWS Systems Manager Hybrid Activations.
3. For other third-party, long-lived secrets that require the use of the rotate strategy, integrate Secrets Manager into your code to retrieve third-party secrets at runtime.
   a. The CodeGuru console can automatically create a secret in Secrets Manager using the discovered credentials.
   b. Integrate secret retrieval from Secrets Manager into your application code.

    i. Serverless Lambda functions can use a language-agnostic Lambda extension.
    ii. For EC2 instances or containers, AWS provides example client-side code for retrieving secrets from Secrets Manager in several popular programming languages.
4. Periodically review your code base and re-scan to verify no new secrets have been added to the code.
    a. Consider using a tool such as git-secrets to prevent committing new secrets to your source code repository.
5. Monitor Secrets Manager activity for indications of unexpected usage, inappropriate secret access, or attempts to delete secrets.
6. Reduce human exposure to credentials. Restrict access to read, write, and modify credentials to an IAM role dedicated for this purpose, and only provide access to assume the role to a small subset of operational users.

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC02-BP05 Audit and rotate credentials periodically

Related documents:

- Getting Started with AWS Secrets Manager
- Identity Providers and Federation
- Amazon CodeGuru Introduces Secrets Detector
- How AWS Secrets Manager uses AWS Key Management Service
- Secret encryption and decryption in Secrets Manager
- Secrets Manager blog entries
- Amazon RDS announces integration with AWS Secrets Manager

Related videos:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Find Hard-Coded Secrets Using Amazon CodeGuru Secrets Detector

- Securing Secrets for Hybrid Workloads Using AWS Secrets Manager

Related workshops:

- Store, retrieve, and manage sensitive credentials in AWS Secrets Manager
- AWS Systems Manager Hybrid Activations

SEC02-BP04 Rely on a centralized identity provider

For workforce identities (employees and contractors), rely on an identity provider that allows you to manage identities in a centralized place. This makes it easier to manage access across multiple applications and systems, because you are creating, assigning, managing, revoking, and auditing access from a single location.

Desired outcome: You have a centralized identity provider where you centrally manage workforce users, authentication policies (such as requiring multi-factor authentication (MFA)), and authorization to systems and applications (such as assigning access based on a user's group membership or attributes). Your workforce users sign in to the central identity provider and federate (single sign-on) to internal and external applications, removing the need for users to remember multiple credentials. Your identity provider is integrated with your human resources (HR) systems so that personnel changes are automatically synchronized to your identity provider. For example, if someone leaves your organization, you can automatically revoke access to federated applications and systems (including AWS). You have enabled detailed audit logging in your identity provider and are monitoring these logs for unusual user behavior.

Common anti-patterns:

- You do not use federation and single-sign on. Your workforce users create separate user accounts and credentials in multiple applications and systems.
- You have not automated the lifecycle of identities for workforce users, such as by integrating your identity provider with your HR systems. When a user leaves your organization or changes roles, you follow a manual process to delete or update their records in multiple applications and systems.

Benefits of establishing this best practice: By using a centralized identity provider, you have a single place to manage workforce user identities and policies, the ability to assign access to applications to users and groups, and the ability to monitor user sign-in activity. By integrating

with your human resources (HR) systems, when a user changes roles, these changes are synchronized to the identity provider and automatically updates their assigned applications and permissions. When a user leaves your organization, their identity is automatically disabled in the identity provider, revoking their access to federated applications and systems.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Guidance for workforce users accessing AWS Workforce users like employees and contractors in your organization may require access to AWS using the AWS Management Console or AWS Command Line Interface (AWS CLI) to perform their job functions. You can grant AWS access to your workforce users by federating from your centralized identity provider to AWS at two levels: direct federation to each AWS account or federating to multiple accounts in your AWS organization.

To federate your workforce users directly with each AWS account, you can use a centralized identity provider to federate to AWS Identity and Access Management in that account. The flexibility of IAM allows you to enable a separate SAML 2.0 or an Open ID Connect (OIDC) Identity Provider for each AWS account and use federated user attributes for access control. Your workforce users will use their web browser to sign in to the identity provider by providing their credentials (such as passwords and MFA token codes). The identity provider issues a SAML assertion to their browser that is submitted to the AWS Management Console sign in URL to allow the user to single sign-on to the AWS Management Console by assuming an IAM Role. Your users can also obtain temporary AWS API credentials for use in the AWS CLI or AWS SDKs from AWS STS by assuming the IAM role using a SAML assertion from the identity provider.

To federate your workforce users with multiple accounts in your AWS organization, you can use *AWS IAM Identity Center* to centrally manage access for your workforce users to AWS accounts and applications. You enable Identity Center for your organization and configure your identity source. IAM Identity Center provides a default identity source directory which you can use to manage your users and groups. Alternatively, you can choose an external identity source by *connecting to your external identity provider* using SAML 2.0 and automatically provisioning users and groups using SCIM, or *connecting to your Microsoft AD Directory* using Directory Service. Once an identity source is configured, you can assign access to users and groups to AWS accounts by defining least-privilege policies in your permission sets. Your workforce users can authenticate through your central identity provider to sign in to the AWS access portal and single-sign on to the AWS accounts and cloud applications assigned to them. Your users can configure the AWS CLI v2 to authenticate with Identity Center and get credentials to run AWS CLI commands. Identity Center also allows

single-sign on access to AWS applications such as Amazon SageMaker AI Studio and AWS IoT Sitewise Monitor portals.

After you follow the preceding guidance, your workforce users will no longer need to use IAM users and groups for normal operations when managing workloads on AWS. Instead, your users and groups are managed outside of AWS and users are able to access AWS resources as a *federated identity*. Federated identities use the groups defined by your centralized identity provider. You should identify and remove IAM groups, IAM users, and long-lived user credentials (passwords and access keys) that are no longer needed in your AWS accounts. You can find unused credentials using IAM credential reports, delete the corresponding IAM users and delete IAM groups. You can apply a Service Control Policy (SCP) to your organization that helps prevent the creation of new IAM users and groups, enforcing that access to AWS is via federated identities.

> **Note**
>
> You are responsible for handling the rotation of SCIM access tokens as described in the Automatic provisioning documentation. Additionally, you are responsible for rotating the certificates supporting your identity federation.

Guidance for users of your applications You can manage the identities of users of your applications, such as a mobile app, using *Amazon Cognito* as your centralized identity provider. Amazon Cognito enables authentication, authorization, and user management for your web and mobile apps. Amazon Cognito provides an identity store that scales to millions of users, supports social and enterprise identity federation, and offers advanced security features to help protect your users and business. You can integrate your custom web or mobile application with Amazon Cognito to add user authentication and access control to your applications in minutes. Built on open identity standards such as SAML and Open ID Connect (OIDC), Amazon Cognito supports various compliance regulations and integrates with frontend and backend development resources.

Implementation steps

Steps for workforce users accessing AWS

- Federate your workforce users to AWS using a centralized identity provider using one of the following approaches:
  - Use IAM Identity Center to enable single sign-on to multiple AWS accounts in your AWS organization by federating with your identity provider.

  - Use IAM to connect your identity provider directly to each AWS account, enabling federated fine-grained access.
- Identify and remove IAM users and groups that are replaced by federated identities.

Steps for users of your applications

- Use Amazon Cognito as a centralized identity provider towards your applications.
- Integrate your custom applications with Amazon Cognito using OpenID Connect and OAuth. You can develop your custom applications using the Amplify libraries that provide simple interfaces to integrate with a variety of AWS services, such as Amazon Cognito for authentication.

Resources

Related best practices:

- SEC02-BP06 Employ user groups and attributes
- SEC03-BP02 Grant least privilege access
- SEC03-BP06 Manage access based on lifecycle

Related documents:

- Identity federation in AWS
- Security best practices in IAM
- AWS Identity and Access Management Best practices
- Getting started with IAM Identity Center delegated administration
- How to use customer managed policies in IAM Identity Center for advanced use cases
- AWS CLI v2: IAM Identity Center credential provider

Related videos:

- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2018: Mastering Identity at Every Layer of the Cake

Related examples:

- Workshop: Using AWS IAM Identity Center to achieve strong identity management

Related tools:

- AWS Security Competency Partners: Identity and Access Management
- saml2aws

SEC02-BP05 Audit and rotate credentials periodically

Audit and rotate credentials periodically to limit how long the credentials can be used to access your resources. Long-term credentials create many risks, and these risks can be reduced by rotating long-term credentials regularly.

Desired outcome: Implement credential rotation to help reduce the risks associated with long-term credential usage. Regularly audit and remediate non-compliance with credential rotation policies.

Common anti-patterns:

- Not auditing credential use.
- Using long-term credentials unnecessarily.
- Using long-term credentials and not rotating them regularly.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When you cannot rely on temporary credentials and require long-term credentials, audit credentials to verify that defined controls like multi-factor authentication (MFA) are enforced, rotated regularly, and have the appropriate access level.

Periodic validation, preferably through an automated tool, is necessary to verify that the correct controls are enforced. For human identities, you should require users to change their passwords periodically and retire access keys in favor of temporary credentials. As you move from AWS Identity and Access Management (IAM) users to centralized identities, you can generate a credential report to audit your users.

We also recommend that you enforce and monitor MFA in your identity provider. You can set up AWS Config Rules, or use AWS Security Hub CSPM Security Standards, to monitor if users have configured MFA. Consider using IAM Roles Anywhere to provide temporary credentials for machine identities. In situations when using IAM roles and temporary credentials is not possible, frequent auditing and rotating access keys is necessary.

Implementation steps

- **Regularly audit credentials:** Auditing the identities that are configured in your identity provider and IAM helps verify that only authorized identities have access to your workload. Such identities can include, but are not limited to, IAM users, AWS IAM Identity Center users, Active Directory users, or users in a different upstream identity provider. For example, remove people that leave the organization, and remove cross-account roles that are no longer required. Have a process in place to periodically audit permissions to the services accessed by an IAM entity. This helps you identify the policies you need to modify to remove any unused permissions. Use credential reports and AWS Identity and Access Management Access Analyzer to audit IAM credentials and permissions. You can use Amazon CloudWatch to set up alarms for specific API calls called within your AWS environment. Amazon GuardDuty can also alert you to unexpected activity, which might indicate overly permissive access or unintended access to IAM credentials.
- **Rotate credentials regularly:** When you are unable to use temporary credentials, rotate long-term IAM access keys regularly (maximum every 90 days). If an access key is unintentionally disclosed without your knowledge, this limits how long the credentials can be used to access your resources. For information about rotating access keys for IAM users, see Rotating access keys.
- **Review IAM permissions:** To improve the security of your AWS account, regularly review and monitor each of your IAM policies. Verify that policies adhere to the principle of least privilege.
- **Consider automating IAM resource creation and updates:** IAM Identity Center automates many IAM tasks, such as role and policy management. Alternatively, AWS CloudFormation can be used to automate the deployment of IAM resources, including roles and policies, to reduce the chance of human error because the templates can be verified and version controlled.
- **Use IAM Roles Anywhere to replace IAM users for machine identities:** IAM Roles Anywhere allows you to use roles in areas that you traditionally could not, such as on-premise servers. IAM Roles Anywhere uses a trusted X.509 certificate to authenticate to AWS and receive temporary credentials. Using IAM Roles Anywhere avoids the need to rotate these credentials, as long-term credentials are no longer stored in your on-premises environment. Please note that you will need to monitor and rotate the X.509 certificate as it approaches expiration.

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC02-BP03 Store and use secrets securely

Related documents:

- Getting Started with AWS Secrets Manager
- IAM Best Practices
- Identity Providers and Federation
- Security Partner Solutions: Access and Access Control
- Temporary Security Credentials
- Getting credential reports for your AWS account

Related videos:

- Best Practices for Managing, Retrieving, and Rotating Secrets at Scale
- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC02-BP06 Employ user groups and attributes

Defining permissions according to user groups and attributes helps reduce the number and complexity of policies, making it simpler to achieve the principle of least privilege. You can use user groups to manage the permissions for many people in one place based on the function they perform in your organization. Attributes, such as department, project, or location, can provide an additional layer of permission scope when people perform a similar function but for different subsets of resources.

Desired outcome: You can apply changes in permissions based on function to all users who perform that function. Group membership and attributes govern user permissions, reducing the need to manage permissions at the individual user level. The groups and attributes you define in your identity provider (IdP) are propagated automatically to your AWS environments.

Common anti-patterns:

- Managing permissions for individual users and duplicating across many users.
- Defining groups at too high a level, granting overly-broad permissions.
- Defining groups at too granular a level, creating duplication and confusion about membership.
- Using groups with duplicate permissions across subsets of resources when attributes can be used instead.
- Not managing groups, attributes, and memberships through a standardized identity provider integrated with your AWS environments.
- Using role chaining when using AWS IAM Identity Center sessions

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS permissions are defined in documents called *policies* that are associated with a principal, such as a user, group, role, or resource. You can scale permissions management by organizing permissions assignments (group, permissions, account) based on job-function, workload, and SDLC environment. For your workforce, this allows you to define groups based on the function your users perform for your organization, rather than based on the resources being accessed. For example, a WebAppDeveloper group may have a policy attached for configuring services like Amazon CloudFront within a development account. An `AutomationDeveloper` group may have some overlapping permissions with the `WebAppDeveloper` group. These common permissions can be captured in a separate policy and associated with both groups, rather than having users from both functions belong to a `CloudFrontAccess` group.

In addition to groups, you can use *attributes* to further scope access. For example, you may have a Project attribute for users in your `WebAppDeveloper` group to scope access to resources specific to their project. Using this technique removes the need to have different groups for application developers working on different projects if their permissions are otherwise the same. The way you refer to attributes in permission policies is based on their source, whether they are defined as part of your federation protocol (such as SAML, OIDC, or SCIM), as custom SAML assertions, or set within IAM Identity Center.

Implementation steps

1. Establish where you will define groups and attributes:

   a. Following the guidance in SEC02-BP04 Rely on a centralized identity provider, you can determine whether you need to define groups and attributes within your identity provider, within IAM Identity Center, or using IAM user groups in a specific account.

2. Define groups:

   a. Determine your groups based on function and scope of access required. Consider using a hierarchical structure or naming conventions to organize groups effectively.

   b. If defining within IAM Identity Center, create groups and associate the desired level of access using permission sets.

   c. If defining within an external identity provider, determine if the provider supports the SCIM protocol and consider enabling automatic provisioning within IAM Identity Center. This capability synchronizes the creation, membership, and deletion of groups between your provider and IAM Identity Center.

3. Define attributes:

   a. If you use an external identity provider, both the SCIM and SAML 2.0 protocols provide certain attributes by default. Additional attributes can be defined and passed using SAML assertions with the `https://aws.amazon.com/SAML/Attributes/PrincipalTag` attribute name. Refer to your identity provider's documentation for guidance on defining and configuring custom attributes.

   b. If you define roles within IAM Identity Center, enable the attribute-based access control (ABAC) feature, and define attributes as desired. Consider attributes that align with your organization's structure or resource tagging strategy.

If you require IAM role chaining from IAM Roles assumed through IAM Identity center, values like `source-identity` and `principal-tags` will not propagate. For more detail, see Enable and configure attributes for access control.

1. Scope permissions based on groups and attributes:

   a. Consider including conditions in your permission policies that compare the attributes of your principal with the attributes of the resources being accessed. For example, you can define a condition to allow access to a resource only if the value of a `PrincipalTag` condition key matches the value of a `ResourceTag` key of the same name.

   b. When defining ABAC policies, follow the guidance in the ABAC authorization best practices and examples.

c. Regularly review and update your group and attribute structure as your organization's needs evolve to ensure optimal permissions management.

Resources

Related best practices:

- SEC02-BP04 Rely on a centralized identity provider
- SEC03-BP02 Grant least privilege access
- COST02-BP04 Implement groups and roles

Related documents:

- IAM Best Practices
- Manage Identities in IAM Identity Center
- What Is ABAC for AWS?
- ABAC In IAM Identity Center
- ABAC Policy Examples

Related videos:

- Managing user permissions at scale with AWS IAM Identity Center
- Mastering identity at every layer of the cake

SEC 3. How do you manage permissions for people and machines?

Manage permissions to control access to human and machine identities that require access to AWS and your workloads. Permissions allow you to control who can access what, and under what conditions. By setting permissions to specific human and machine identities, you grant them access to specific service actions on specific resources. Additionally, you can specify conditions that must be true for access to be granted.

Best practices

- SEC03-BP01 Define access requirements
- SEC03-BP02 Grant least privilege access

- SEC03-BP03 Establish emergency access process
- SEC03-BP04 Reduce permissions continuously
- SEC03-BP05 Define permission guardrails for your organization
- SEC03-BP06 Manage access based on lifecycle
- SEC03-BP07 Analyze public and cross-account access
- SEC03-BP08 Share resources securely within your organization
- SEC03-BP09 Share resources securely with a third party

SEC03-BP01 Define access requirements

Each component or resource of your workload needs to be accessed by administrators, end users, or other components. Have a clear definition of who or what should have access to each component, choose the appropriate identity type and method of authentication and authorization.

Common anti-patterns:

- Hard-coding or storing secrets in your application.
- Granting custom permissions for each user.
- Using long-lived credentials.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Each component or resource of your workload needs to be accessed by administrators, end users, or other components. Have a clear definition of who or what should have access to each component, choose the appropriate identity type and method of authentication and authorization.

Regular access to AWS accounts within the organization should be provided using federated access or a centralized identity provider. You should also centralize your identity management and ensure that there is an established practice to integrate AWS access to your employee access lifecycle. For example, when an employee changes to a job role with a different access level, their group membership should also change to reflect their new access requirements.

When defining access requirements for non-human identities, determine which applications and components need access and how permissions are granted. Using IAM roles built with the least

privilege access model is a recommended approach. AWS Managed policies provide predefined IAM policies that cover most common use cases.

AWS services, such as AWS Secrets Manager and AWS Systems Manager Parameter Store, can help decouple secrets from the application or workload securely. In Secrets Manager, you can establish automatic rotation for your credentials. You can use Systems Manager to reference parameters in your scripts, commands, SSM documents, configuration, and automation workflows by using the unique name that you specified when you created the parameter.

You can use AWS IAM Roles Anywhere to obtain temporary security credentials in IAM for workloads that run outside of AWS. Your workloads can use the same IAM policies and IAM roles that you use with AWS applications to access AWS resources.

Where possible, prefer short-term temporary credentials over long-term static credentials. For scenarios in which you need users with programmatic access and long-term credentials, use access key last used information to rotate and remove access keys.

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
IAM	(Recommended) Use console credentials as temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Login for AWS local development in the *AWS Command Line Interface User Guide*. • For AWS SDKs, see Login for AWS local development in the *AWS SDKs and Tools Reference Guide*.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. • For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Resources

Related documents:

- Attribute-based access control (ABAC)
- AWS IAM Identity Center
- IAM Roles Anywhere
- AWS Managed policies for IAM Identity Center
- AWS IAM policy conditions
- IAM use cases
- Remove unnecessary credentials
- Working with Policies
- How to control access to AWS resources based on AWS account, OU, or organization

- Identify, arrange, and manage secrets easily using enhanced search in AWS Secrets Manager

Related videos:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- Streamlining identity and access management for innovation

SEC03-BP02 Grant least privilege access

Grant only the access that users require to perform specific actions on specific resources under specific conditions. Use group and identity attributes to dynamically set permissions at scale, rather than defining permissions for individual users. For example, you can allow a group of developers access to manage only resources for their project. This way, if a developer leaves the project, their access is automatically revoked without changing the underlying access policies.

Desired outcome: Users have only the minimum permissions required for their specific job functions. You use separate AWS accounts to isolate developers from production environments. When developers need to access production environments for specific tasks, they are granted limited and controlled access only for the duration of those tasks. Their production access is immediately revoked after they complete the necessary work. You conduct regular reviews of permissions and promptly revoke them when no longer needed, such as when a user changes roles or leaves the organization. You restrict administrator privileges to a small, trusted group to reduce risk exposure. You give machine or system accounts only the minimum permissions required to perform their intended tasks.

Common anti-patterns:

- By default, you grant users administrator permissions.
- You use the root user account for daily activities.
- You create overly permissive policies without proper scoping.
- Your permissions reviews are infrequent, which leads to permissions creep.
- You rely solely on attribute-based access control for environment isolation or permissions management.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The principle of least privilege states that identities should only be permitted to perform the smallest set of actions necessary to fulfill a specific task. This balances usability, efficiency, and security. Operating under this principle helps limit unintended access and helps track who has access to what resources. IAM users and roles have no permissions by default. The root user has full access by default and should be tightly controlled, monitored, and used only for tasks that require root access.

IAM policies are used to explicitly grant permissions to IAM roles or specific resources. For example, identity-based policies can be attached to IAM groups, while S3 buckets can be controlled by resource-based policies.

When you create an IAM policy, you can specify the service actions, resources, and conditions that must be true for AWS to allow or deny access. AWS supports a variety of conditions to help you scope down access. For example, by using the PrincipalOrgID condition key, you can deny actions if the requestor isn't a part of your AWS Organization.

You can also control requests that AWS services make on your behalf, such as AWS CloudFormation creating an AWS Lambda function, using the CalledVia condition key. You can layer different policy types to establish defense-in-depth and limit the overall permissions of your users. You can also restrict what permissions can be granted and under what conditions. For example, you can allow your workload teams to create their own IAM policies for systems they build, but only if they apply a Permission Boundary to limit the maximum permissions they can grant.

Implementation steps

- **Implement least privilege policies**: Assign access policies with least privilege to IAM groups and roles to reflect the user's role or function that you have defined.
- **Isolate development and production environments through separate AWS accounts**: Use separate AWS accounts for development and production environments, and control access between them using service control policies, resource policies, and identity policies.
- **Base policies on API usage**: One way to determine the needed permissions is to review AWS CloudTrail logs. You can use this review to create permissions tailored to the actions that the user actually performs within AWS. IAM Access Analyzer can automatically generate an IAM policy based on access activity. You can use IAM Access Advisor at the organization or account level to track the last accessed information for a particular policy.
- **Consider using AWS managed policies for job functions**: When you begin to create fine-grained permissions policies, it can be helpful to use AWS managed policies for common job roles, such

as billing, database administrators, and data scientists. These policies can help narrow the access that users have while you determine how to implement the least privilege policies.

- **Remove unnecessary permissions:** Detect and remove unused IAM entities, credentials, and permissions to achieve the principle of least privilege. You can use IAM Access Analyzer to identify external and unused access, and IAM Access Analyzer policy generation can help fine-tune permissions policies.
- **Ensure that users have limited access to production environments:** Users should only have access to production environments with a valid use case. After the user performs the specific tasks that required production access, access should be revoked. Limiting access to production environments helps prevent unintended production-impacting events and lowers the scope of impact of unintended access.
- **Consider permissions boundaries:** A permissions boundary is a feature for using a managed policy that sets the maximum permissions that an identity-based policy can grant to an IAM entity. An entity's permissions boundary allows it to perform only the actions that are allowed by both its identity-based policies and its permissions boundaries.
- **Refine access using attribute-based access control and resource tags** Attribute-based access control (ABAC) using resource tags can be used to refine permissions when supported. You can use an ABAC model that compares principal tags to resource tags to refine access based on custom dimensions you define. This approach can simplify and reduce the number of permission policies in your organization.
  - It is recommended that ABAC only be used for access control when both the principals and resources are owned by your AWS Organization. External parties may use the same tag names and values as your organization for their own principals and resources. If you rely solely on these name-value pairs for granting access to external party principals or resources, you may provide unintended permissions.
- **Use service control policies for AWS Organizations:** Service control policies centrally control the maximum available permissions for member accounts in your organization. Importantly, you can use service control policies to restrict root user permissions in member accounts. Also consider using AWS Control Tower, which provides prescriptive managed controls that enrich AWS Organizations. You can also define your own controls within Control Tower.
- **Establish a user lifecycle policy for your organization:** User lifecycle policies define tasks to perform when users are onboarded onto AWS, change job role or scope, or no longer need access to AWS. Perform permission reviews during each step of a user's lifecycle to verify that permissions are properly restrictive and to avoid permissions creep.

- **Establish a regular schedule to review permissions and remove any unneeded permissions:** You should regularly review user access to verify that users do not have overly permissive access. AWS Config and IAM Access Analyzer can help during user permissions audits.
- **Establish a job role matrix:** A job role matrix visualizes the various roles and access levels required within your AWS footprint. With a job role matrix, you can define and separate permissions based on user responsibilities within your organization. Use groups instead of applying permissions directly to individual users or roles.

Resources

Related documents:

- Grant least privilege
- Permissions boundaries for IAM entities
- Techniques for writing least privilege IAM policies
- IAM Access Analyzer makes it easier to implement least privilege permissions by generating IAM policies based on access activity
- Delegate permission management to developers by using IAM permissions boundaries
- Refining Permissions using last accessed information
- IAM policy types and when to use them
- Testing IAM policies with the IAM policy simulator
- Guardrails in AWS Control Tower
- Zero Trust architectures: An AWS perspective
- How to implement the principle of least privilege with CloudFormation StackSets
- Attribute-based access control (ABAC)
- Reducing policy scope by viewing user activity
- View role access
- Use Tagging to Organize Your Environment and Drive Accountability
- AWS Tagging Strategies
- Tagging AWS resources

Related videos:

- Next-generation permissions management

- Zero Trust: An AWS perspective

SEC03-BP03 Establish emergency access process

Create a process that allows for emergency access to your workloads in the unlikely event of an issue with your centralized identity provider.

You must design processes for different failure modes that may result in an emergency event. For example, under normal circumstances, your workforce users federate to the cloud using a centralized identity provider (SEC02-BP04) to manage their workloads. However, if your centralized identity provider fails, or the configuration for federation in the cloud is modified, then your workforce users may not be able to federate into the cloud. An emergency access process allows authorized administrators to access your cloud resources through alternate means (such as an alternate form of federation or direct user access) to fix issues with your federation configuration or your workloads. The emergency access process is used until the normal federation mechanism is restored.

Desired outcome:

- You have defined and documented the failure modes that count as an emergency: consider your normal circumstances and the systems your users depend on to manage their workloads. Consider how each of these dependencies can fail and cause an emergency situation. You may find the questions and best practices in the Reliability pillar useful to identify failure modes and architect more resilient systems to minimize the likelihood of failures.
- You have documented the steps that must be followed to confirm a failure as an emergency. For example, you can require your identity administrators to check the status of your primary and standby identity providers and, if both are unavailable, declare an emergency event for identity provider failure.
- You have defined an emergency access process specific to each type of emergency or failure mode. Being specific can reduce the temptation on the part of your users to overuse a general process for all types of emergencies. Your emergency access processes describe the circumstances under which each process should be used, and conversely situations where the process should not be used and points to alternate processes that may apply.
- Your processes are well-documented with detailed instructions and playbooks that can be followed quickly and efficiently. Remember that an emergency event can be a stressful time for your users and they may be under extreme time pressure, so design your process to be as simple as possible.

Common anti-patterns:

- You do not have well-documented and well-tested emergency access processes. Your users are unprepared for an emergency and follow improvised processes when an emergency event arises.
- Your emergency access processes depend on the same systems (such as a centralized identity provider) as your normal access mechanisms. This means that the failure of such a system may impact both your normal and emergency access mechanisms and impair your ability to recover from the failure.
- Your emergency access processes are used in non-emergency situations. For example, your users frequently misuse emergency access processes as they find it easier to make changes directly than submit changes through a pipeline.
- Your emergency access processes do not generate sufficient logs to audit the processes, or the logs are not monitored to alert for potential misuse of the processes.

Benefits of establishing this best practice:

- By having well-documented and well-tested emergency access processes, you can reduce the time taken by your users to respond to and resolve an emergency event. This can result in less downtime and higher availability of the services you provide to your customers.
- You can track each emergency access request and detect and alert on unauthorized attempts to misuse the process for non-emergency events.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

This section provides guidance for creating emergency access processes for several failure modes related to workloads deployed on AWS, starting with common guidance that applies to all failure modes and followed by specific guidance based on the type of failure mode.

Common guidance for all failure modes

Consider the following as you design an emergency access process for a failure mode:

- Document the pre-conditions and assumptions of the process: when the process should be used and when it should not be used. It helps to detail the failure mode and document assumptions, such as the state of other related systems. For example, the process for the Failure Mode 2

assumes that the identity provider is available, but the configuration on AWS is modified or has expired.

- Pre-create resources needed by the emergency access process (SEC10-BP05). For example, pre-create the emergency access AWS account with IAM users and roles, and the cross-account IAM roles in all the workload accounts. This verifies that these resources are ready and available when an emergency event happens. By pre-creating resources, you do not have a dependency on AWS control plane APIs (used to create and modify AWS resources) that may be unavailable in an emergency. Further, by pre-creating IAM resources, you do not need to account for potential delays due to eventual consistency.
- Include emergency access processes as part of your incident management plans (SEC10-BP02). Document how emergency events are tracked and communicated to others in your organization such as peer teams, your leadership, and, when applicable, externally to your customers and business partners.
- Define the emergency access request process in your existing service request workflow system if you have one. Typically, such workflow systems allow you to create intake forms to collect information about the request, track the request through each stage of the workflow, and add both automated and manual approval steps. Relate each request with a corresponding emergency event tracked in your incident management system. Having a uniform system for emergency accesses allows you to track those requests in a single system, analyze usage trends, and improve your processes.
- Verify that your emergency access processes can only be initiated by authorized users and require approvals from the user's peers or management as appropriate. The approval process should operate effectively both inside and outside business hours. Define how requests for approval allow secondary approvers if the primary approvers are unavailable and are escalated up your management chain until approved.
- Implement robust logging, monitoring, and alerting mechanisms for the emergency access process and mechanisms. Generate detailed audit logs for all successful and failed attempts to gain emergency access. Correlate the activity with ongoing emergency events from your incident management system, and initiate alerts when actions occur outside of expected time periods or when the emergency access account is used during normal operations. The emergency access account should only be accessed during emergencies, as break-glass procedures can be considered a backdoor. Integrate with your security information and event management (SIEM) tool or AWS Security Hub CSPM to report and audit all activities during the emergency access period. Upon returning to normal operations, automatically rotate the emergency access credentials, and notify the relevant teams.

- Test emergency access processes periodically to verify that the steps are clear and grant the correct level of access quickly and efficiently. Your emergency access processes should be tested as part of incident response simulations (SEC10-BP07) and disaster recovery tests (REL13-BP03).

Failure Mode 1: Identity provider used to federate to AWS is unavailable

As described in SEC02-BP04 Rely on a centralized identity provider, we recommend relying on a centralized identity provider to federate your workforce users to grant access to AWS accounts. You can federate to multiple AWS accounts in your AWS organization using IAM Identity Center, or you can federate to individual AWS accounts using IAM. In both cases, workforce users authenticate with your centralized identity provider before being redirected to an AWS sign-in endpoint to single sign-on.

In the unlikely event that your centralized identity provider is unavailable, your workforce users can't federate to AWS accounts or manage their workloads. In this emergency event, you can provide an emergency access process for a small set of administrators to access AWS accounts to perform critical tasks that cannot wait until your centralized identity providers are back online. For example, your identity provider is unavailable for 4 hours and during that period you need to modify the upper limits of an Amazon EC2 Auto Scaling group in a Production account to handle an unexpected spike in customer traffic. Your emergency administrators should follow the emergency access process to gain access to the specific production AWS account and make the necessary changes.

The emergency access process relies on a pre-created emergency access AWS account that is used solely for emergency access and has AWS resources (such as IAM roles and IAM users) to support the emergency access process. During normal operations, no one should access the emergency access account and you must monitor and alert on the misuse of this account (for more detail, see the preceding Common guidance section).

The emergency access account has emergency access IAM roles with permissions to assume cross-account roles in the AWS accounts that require emergency access. These IAM roles are pre-created and configured with trust policies that trust the emergency account's IAM roles.

The emergency access process can use one of the following approaches:

- You can pre-create a set of IAM users for your emergency administrators in the emergency access account with associated strong passwords and MFA tokens. These IAM users have permissions to assume the IAM roles that then allow cross-account access to the AWS account where emergency access is required. We recommend creating as few such users as possible and assigning each

user to a single emergency administrator. During an emergency, an emergency administrator user signs into the emergency access account using their password and MFA token code, switches to the emergency access IAM role in the emergency account, and finally switches to the emergency access IAM role in the workload account to perform the emergency change action. The advantage of this approach is that each IAM user is assigned to one emergency administrator and you can know which user signed-in by reviewing CloudTrail events. The disadvantage is that you have to maintain multiple IAM users with their associated long-lived passwords and MFA tokens.

- You can use the emergency access AWS account root user to sign into the emergency access account, assume the IAM role for emergency access, and assume the cross-account role in the workload account. We recommend setting a strong password and multiple MFA tokens for the root user. We also recommend storing the password and the MFA tokens in a secure enterprise credential vault that enforces strong authentication and authorization. You should secure the password and MFA token reset factors: set the email address for the account to an email distribution list that is monitored by your cloud security administrators, and the phone number of the account to a shared phone number that is also monitored by security administrators. The advantage of this approach is that there is one set of root user credentials to manage. The disadvantage is that since this is a shared user, multiple administrators have ability to sign in as the root user. You must audit your enterprise vault log events to identify which administrator checked out the root user password.

Failure Mode 2: Identity provider configuration on AWS is modified or has expired

To allow your workforce users to federate to AWS accounts, you can configure the IAM Identity Center with an external identity provider or create an IAM Identity Provider (SEC02-BP04). Typically, you configure these by importing a SAML meta-data XML document provided by your identity provider. The meta-data XML document includes a X.509 certificate corresponding to a private key that the identity provider uses to sign its SAML assertions.

These configurations on the AWS-side may be modified or deleted by mistake by an administrator. In another scenario, the X.509 certificate imported into AWS may expire and a new meta-data XML with a new certificate has not yet been imported into AWS. Both scenarios can break federation to AWS for your workforce users, resulting in an emergency.

In such an emergency event, you can provide your identity administrators access to AWS to fix the federation issues. For example, your identity administrator uses the emergency access process to sign into the emergency access AWS account, switches to a role in the Identity Center administrator

account, and updates the external identity provider configuration by importing the latest SAML meta-data XML document from your identity provider to re-enable federation. Once federation is fixed, your workforce users continue to use the normal operating process to federate into their workload accounts.

You can follow the approaches detailed in the previous Failure Mode 1 to create an emergency access process. You can grant least-privilege permissions to your identity administrators to access only the Identity Center administrator account and perform actions on Identity Center in that account.

Failure Mode 3: Identity Center disruption

In the unlikely event of an IAM Identity Center or AWS Region disruption, we recommend that you set up a configuration that you can use to provide temporary access to the AWS Management Console.

The emergency access process uses direct federation from your identity provider to IAM in an emergency account. For detail on the process and design considerations, see Set up emergency access to the AWS Management Console.

Implementation steps

Common steps for all failure modes

- Create an AWS account dedicated to emergency access processes. Pre-create the IAM resources needed in the account such as IAM roles or IAM users, and optionally IAM Identity Providers. Additionally, pre-create cross-account IAM roles in the workload AWS accounts with trust relationships with corresponding IAM roles in the emergency access account. You can use CloudFormation StackSets with AWS Organizations to create such resources in the member accounts in your organization.

- Create AWS Organizations service control policies (SCPs) to deny the deletion and modification of the cross-account IAM roles in the member AWS accounts.
- Enable CloudTrail for the emergency access AWS account and send the trail events to a central S3 bucket in your log collection AWS account. If you are using AWS Control Tower to set up and govern your AWS multi-account environment, then every account you create using AWS Control Tower or enroll in AWS Control Tower has CloudTrail enabled by default and sent to an S3 bucket in a dedicated log archive AWS account.

- Monitor the emergency access account for activity by creating EventBridge rules that match on console login and API activity by the emergency IAM roles. Send notifications to your security operations center when activity happens outside of an ongoing emergency event tracked in your incident management system.

Additional steps for Failure Mode 1: Identity provider used to federate to AWS is unavailable and Failure Mode 2: Identity provider configuration on AWS is modified or has expired

- Pre-create resources depending on the mechanism you choose for emergency access:
  - **Using IAM users:** pre-create the IAM users with strong passwords and associated MFA devices.
  - **Using the emergency account root user:** configure the root user with a strong password and store the password in your enterprise credential vault. Associate multiple physical MFA devices with the root user and store the devices in locations that can be accessed quickly by members of your emergency administrator team.

Additional steps for Failure Mode 3: Identity center disruption

- As detailed in Set up emergency access to the AWS Management Console, in the emergency access AWS account, create an IAM Identity Provider to enable direct SAML federation from your identity provider.
- Create emergency operations groups in your IdP with no members.
- Create IAM roles corresponding to the emergency operations groups in the emergency access account.

Resources

Related Well-Architected best practices:

- SEC02-BP04 Rely on a centralized identity provider
- SEC03-BP02 Grant least privilege access
- SEC10-BP02 Develop incident management plans
- SEC10-BP07 Run game days

Related documents:

- Set up emergency access to the AWS Management Console

- Enabling SAML 2.0 federated users to access the AWS Management Console
- Break glass access

Related videos:

- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

Related examples:

- AWS Break Glass Role
- AWS customer playbook framework
- AWS incident response playbook samples

SEC03-BP04 Reduce permissions continuously

As your teams determine what access is required, remove unneeded permissions and establish review processes to achieve least privilege permissions. Continually monitor and remove unused identities and permissions for both human and machine access.

Desired outcome: Permission policies should adhere to the least privilege principle. As job duties and roles become better defined, your permission policies need to be reviewed to remove unnecessary permissions. This approach lessens the scope of impact should credentials be inadvertently exposed or otherwise accessed without authorization.

Common anti-patterns:

- Defaulting to granting users administrator permissions.
- Creating policies that are overly permissive, but without full administrator privileges.
- Keeping permission policies after they are no longer needed.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As teams and projects are just getting started, permissive permission policies might be used to inspire innovation and agility. For example, in a development or test environment, developers

can be given access to a broad set of AWS services. We recommend that you evaluate access continuously and restrict access to only those services and service actions that are necessary to complete the current job. We recommend this evaluation for both human and machine identities. Machine identities, sometimes called system or service accounts, are identities that give AWS access to applications or servers. This access is especially important in a production environment, where overly permissive permissions can have a broad impact and potentially expose customer data.

AWS provides multiple methods to help identify unused users, roles, permissions, and credentials. AWS can also help analyze access activity of IAM users and roles, including associated access keys, and access to AWS resources such as objects in Amazon S3 buckets. AWS Identity and Access Management Access Analyzer policy generation can assist you in creating restrictive permission policies based on the actual services and actions a principal interacts with. Attribute-based access control (ABAC) can help simplify permissions management, as you can provide permissions to users using their attributes instead of attaching permissions policies directly to each user.

Implementation steps

- **Use AWS Identity and Access Management Access Analyzer:** IAM Access Analyzer helps identify resources in your organization and accounts, such as Amazon Simple Storage Service (Amazon S3) buckets or IAM roles that are shared with an external entity.
- **Use IAM Access Analyzer policy generation:** IAM Access Analyzer policy generation helps you create fine-grained permission policies based on an IAM user or role's access activity.
- **Test permissions across lower environments before production:** Start by using the less critical sandbox and development environments to test the permissions required for various job functions using IAM Access Analyzer. Then, progressively tighten and validate these permissions across the testing, quality assurance, and staging environments before applying them to production. The lower environments can have more relaxed permissions initially, as service control policies (SCPs) enforce guardrails by limiting the maximum permissions granted.
- **Determine an acceptable timeframe and usage policy for IAM users and roles:** Use the last accessed timestamp to identify unused users and roles and remove them. Review service and action last accessed information to identify and scope permissions for specific users and roles. For example, you can use last accessed information to identify the specific Amazon S3 actions that your application role requires and restrict the role's access to only those actions. Last accessed information features are available in the AWS Management Console and programmatically allow you to incorporate them into your infrastructure workflows and automated tools.

- **Consider logging data events in AWS CloudTrail:** By default, CloudTrail does not log data events such as Amazon S3 object-level activity (for example, `GetObject` and `DeleteObject`) or Amazon DynamoDB table activities (for example, `PutItem` and `DeleteItem`). Consider using logging for these events to determine what users and roles need access to specific Amazon S3 objects or DynamoDB table items.

Resources

Related documents:

- Grant least privilege
- Remove unnecessary credentials
- What is AWS CloudTrail?
- Working with Policies
- Logging and monitoring DynamoDB
- Using CloudTrail event logging for Amazon S3 buckets and objects
- Getting credential reports for your AWS account

Related videos:

- Become an IAM Policy Master in 60 Minutes or Less
- Separation of Duties, Least Privilege, Delegation, and CI/CD
- AWS re:Inforce 2022 - AWS Identity and Access Management (IAM) deep dive

SEC03-BP05 Define permission guardrails for your organization

Use permission guardrails to reduce the scope of available permissions that can be granted to principals. The permission policy evaluation chain includes your guardrails to determine the *effective permissions* of a principal when making authorization decisions. You can define guardrails using a layer-based approach. Apply some guardrails broadly across your entire organization and apply others granularly to temporary access sessions.

Desired outcome: You have clear isolation of environments using separate AWS accounts. Service control policies (SCPs) are used to define organization-wide permission guardrails. Broader guardrails are set at the hierarchy levels closest to your organization root, and more strict guardrails are set closer to the level of individual accounts.

Where supported, resource policies define the conditions that a principal must satisfy to gain access to a resource. Resource policies also scope down the set of allowable actions, where appropriate. Permission boundaries are placed on principals that manage workload permissions, delegating permission management to individual workload owners.

Common anti-patterns:

- Creating member AWS accounts within an AWS Organization, but not using SCPs to restrict the use and permissions available to their root credentials.
- Assigning permissions based on least privilege, but not placing guardrails on the maximum set of permissions that can be granted.
- Relying on the *implicit deny* foundation of AWS IAM to restrict permissions, trusting that policies will not grant an undesired *explicit allow* permission.
- Running multiple workload environments in the same AWS account, and then relying on mechanisms such as VPCs, tags, or resource policies to enforce permission boundaries.

Benefits of establishing this best practice: Permission guardrails help build confidence that undesired permissions cannot be granted, even when a permission policy attempts to do so. This can simplify defining and managing permissions by reducing the maximum scope of permissions needing consideration.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

We recommend you use a layer-based approach to define permission guardrails for your organization. This approach systematically reduces the maximum set of possible permissions as additional layers are applied. This helps you grant access based on the principle of least privilege, reducing the risk of unintended access due to policy misconfiguration.

The first step to establish permission guardrails is to isolate your workloads and environments into separate AWS accounts. Principals from one account cannot access resources in another account without explicit permission to do so, even when both accounts are in the same AWS organization or under the same organizational unit (OU). You can use OUs to group accounts you want to administer as a single unit.

The next step is to reduce the maximum set of permissions that you can grant to principals within the member accounts of your organization. You can use service control policies (SCPs) for this

purpose, which you can apply to either an OU or an account. SCPs can enforce common access controls, such as restricting access to specific AWS Regions, help prevent resources from being deleted, or disabling potentially risky service actions. SCPs that you apply to the root of your organization only affect its member accounts, not the management account. SCPs only govern the principals within your organization. Your SCPs don't govern principals outside your organization that are accessing your resources.

If you are using AWS Control Tower, you can leverage its controls and landing zones as the foundation for your permission guardrails and multi-account environment. The landing zones provide a pre-configured, secure baseline environment with separate accounts for different workloads and applications. The guardrails enforce mandatory controls around security, operations, and compliance through a combination of Service Control Policies (SCPs), AWS Config rules, and other configurations. However, when using Control Tower guardrails and landing zones alongside custom Organization SCPs, it's crucial to follow the best practices outlined in the AWS documentation to avoid conflicts and ensure proper governance. Refer to the AWS Control Tower guidance for AWS Organizations for detailed recommendations on managing SCPs, accounts, and organizational units (OUs) within a Control Tower environment.

By adhering to these guidelines, you can effectively leverage Control Tower's guardrails, landing zones, and custom SCPs while mitigating potential conflicts and ensuring proper governance and control over your multi-account AWS environment.

A further step is to use IAM resource policies to scope the available actions that you can take on the resources they govern, along with any conditions that the acting principal must meet. This can be as broad as allowing all actions so long as the principal is part of your organization (using the PrincipalOrgId condition key), or as granular as only allowing specific actions by a specific IAM role. You can take a similar approach with conditions in IAM role trust policies. If a resource or role trust policy explicitly names a principal in the same account as the role or resource it governs, that principal does not need an attached IAM policy that grants the same permissions. If the principal is in a different account from the resource, then the principal does need an attached IAM policy that grants those permissions.

Often, a workload team will want to manage the permissions their workload requires. This may require them to create new IAM roles and permission policies. You can capture the maximum scope of permissions the team is allowed to grant in an IAM permission boundary, and associate this document to an IAM role the team can then use to manage their IAM roles and permissions. This approach can provide them the flexibility to complete their work while mitigating risks of having IAM administrative access.

A more granular step is to implement *privileged access management* (PAM) and *temporary elevated access management* (TEAM) techniques. One example of PAM is to require principals to perform multi-factor authentication before taking privileged actions. For more information, see Configuring MFA-protected API access. TEAM requires a solution that manages the approval and timeframe that a principal is allowed to have elevated access. One approach is to temporarily add the principal to the role trust policy for an IAM role that has elevated access. Another approach is to, under normal operation, scope down the permissions granted to a principal by an IAM role using a session policy, and then temporarily lift this restriction during the approved time window. To learn more about solutions that AWS and select partners validated, see Temporary elevated access.

Implementation steps

1. Isolate your workloads and environments into separate AWS accounts.
2. Use SCPs to reduce the maximum set of permissions that can be granted to principals within the member accounts of your organization.
   a. When defining SCPs to reduce the maximum set of permissions that can be granted to principals within your organization's member accounts, you can choose between an *allow list* or *deny list* approach. The allow list strategy explicitly specifies the access that is allowed and implicitly blocks all other access. The deny list strategy explicitly specifies the access that isn't allowed and allows all other access by default. Both strategies have their advantages and trade-offs, and the appropriate choice depends on your organization's specific requirements and risk model. For more detail, see Strategy for using SCPs.
   b. Additionally, review the service control policy examples to understand how to construct SCPs effectively.
3. Use IAM resource policies to scope down and specify conditions for permitted actions on resources. Use conditions in IAM role trust policies to create restrictions on assuming roles.
4. Assign IAM permission boundaries to IAM roles that workload teams can then use to manage their own workload IAM roles and permissions.
5. Evaluate PAM and TEAM solutions based on your needs.

Resources

Related documents:

- Data perimeters on AWS

- Establish permissions guardrails using data perimeters
- Policy evaluation logic

Related examples:

- Service control policy examples

Related tools:

- AWS Solution: Temporary Elevated Access Management
- Validated security partner solutions for TEAM

SEC03-BP06 Manage access based on lifecycle

Monitor and adjust the permissions granted to your principals (users, roles, and groups) throughout their lifecycle within your organization. Adjust group memberships as users change roles, and remove access when a user leaves the organization.

Desired outcome: You monitor and adjust permissions throughout the lifecycle of principals within the organization, reducing risk of unnecessary privileges. You grant appropriate access when you create a user. You modify access as the user's responsibilities change, and you remove access when the user is no longer active or has left the organization. You centrally manage changes to your users, roles, and groups. You use automation to propagate changes to your AWS environments.

Common anti-patterns:

- Granting excessive or broad access privileges to identities upfront, beyond what is initially required.
- Not reviewing and adjusting access privileges as identities' roles and responsibilities change over time.
- Leaving inactive or terminated identities with active access privileges. This increases the risk of unauthorized access.
- Not leveraging automation to manage the lifecycle of identities.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Carefully manage and adjust access privileges that you grant to identities (such as users, roles, groups) throughout their lifecycle. This lifecycle includes the initial onboarding phase, ongoing changes in roles and responsibilities, and eventual offboarding or termination. Proactively manage access based on the stage of the lifecycle to maintain the appropriate access level. Adhere to the principle of least privilege to reduce the risk of excessive or unnecessary access Privileges.

You can manage the lifecycle of IAM users directly within the AWS account, or through federation from your workforce identity provider to AWS IAM Identity Center. For IAM users, you can create, modify, and delete users and their associated permissions within the AWS account. For federated users, you can use IAM Identity Center to manage their lifecycle by synchronizing user and group information from your organization's identity provider using the System for Cross-domain Identity Management (SCIM) protocol.

SCIM is an open standard protocol for automated provisioning and deprovisioning of user identities across different systems. By integrating your identity provider with IAM Identity Center using SCIM, you can automatically synchronize user and group information, helping to validate that access privileges are granted, modified, or revoked based on changes in your organization's authoritative identity source.

As the roles and responsibilities of employees change within your organization, adjust their access privileges accordingly. You can use IAM Identity Center's permission sets to define different job roles or responsibilities and associate them with the appropriate IAM policies and permissions. When an employee's role changes, you can update their assigned permission set to reflect their new responsibilities. Verify that they have the necessary access while adhering to the principle of least privilege.

Implementation steps

1. Define and document an access management lifecycle process, including procedures for granting initial access, periodic reviews, and offboarding.
2. Implement IAM roles, groups, and permissions boundaries to manage access collectively and enforce maximum permissible access levels.
3. Integrate with a federated identity provider (such as Microsoft Active Directory, Okta, Ping Identity) as the authoritative source for user and group information using IAM Identity Center.
4. Use the SCIM protocol to synchronize user and group information from the identity provider into IAM Identity Center's Identity Store.

5. Create permission sets in IAM Identity Center that represent different job roles or responsibilities within your organization. Define the appropriate IAM policies and permissions for each permission set.
6. Implement regular access reviews, prompt access revocation, and continuous improvement of the access management lifecycle process.
7. Provide training and awareness to employees on access management best practices.

Resources

Related best practices:

- SEC02-BP04 Rely on a centralized identity provider

Related documents:

- Manage your identity source
- Manage identities in IAM Identity Center
- Using AWS Identity and Access Management Access Analyzer
- IAM Access Analyzer policy generation

Related videos:

- AWS re:Inforce 2023 - Manage temporary elevated access with AWS IAM Identity Center
- AWS re:Invent 2022 - Simplify your existing workforce access with IAM Identity Center
- AWS re:Invent 2022 - Harness power of IAM policies & rein in permissions w/Access Analyzer

SEC03-BP07 Analyze public and cross-account access

Continually monitor findings that highlight public and cross-account access. Reduce public access and cross-account access to only the specific resources that require this access.

Desired outcome: Know which of your AWS resources are shared and with whom. Continually monitor and audit your shared resources to verify they are shared with only authorized principals.

Common anti-patterns:

- Not keeping an inventory of shared resources.

- Not following a process for approval of cross-account or public access to resources.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

If your account is in AWS Organizations, you can grant access to resources to the entire organization, specific organizational units, or individual accounts. If your account is not a member of an organization, you can share resources with individual accounts. You can grant direct cross-account access using resource-based policies — for example, Amazon Simple Storage Service (Amazon S3) bucket policies — or by allowing a principal in another account to assume an IAM role in your account. When using resource policies, verify that access is only granted to authorized principals. Define a process to approve all resources which are required to be publicly available.

AWS Identity and Access Management Access Analyzer uses provable security to identify all access paths to a resource from outside of its account. It reviews resource policies continuously, and reports findings of public and cross-account access to make it simple for you to analyze potentially broad access. Consider configuring IAM Access Analyzer with AWS Organizations to verify that you have visibility to all your accounts. IAM Access Analyzer also allows you to preview findings before deploying resource permissions. This allows you to validate that your policy changes grant only the intended public and cross-account access to your resources. When designing for multi-account access, you can use trust policies to control in what cases a role can be assumed. For example, you could use the `PrincipalOrgId` condition key to deny an attempt to assume a role from outside your AWS Organizations.

AWS Config can report resources that are misconfigured, and through AWS Config policy checks, can detect resources that have public access configured. Services such as AWS Control Tower and AWS Security Hub CSPM simplify deploying detective controls and guardrails across AWS Organizations to identify and remediate publicly exposed resources. For example, AWS Control Tower has a managed guardrail which can detect if any Amazon EBS snapshots are restorable by AWS accounts.

Implementation steps

- **Consider using AWS Config for AWS Organizations:** AWS Config allows you to aggregate findings from multiple accounts within an AWS Organizations to a delegated administrator account. This provides a comprehensive view, and allows you to deploy AWS Config Rules across accounts to detect publicly accessible resources.

- **Configure AWS Identity and Access Management Access Analyzer:** IAM Access Analyzer helps you identify resources in your organization and accounts, such as Amazon S3 buckets or IAM roles that are shared with an external entity.
- **Use auto-remediation in AWS Config to respond to changes in public access configuration of Amazon S3 buckets:** You can automatically turn on the block public access settings for Amazon S3 buckets.
- **Implement monitoring and alerting to identify if Amazon S3 buckets have become public:** You must have monitoring and alerting in place to identify when Amazon S3 Block Public Access is turned off, and if Amazon S3 buckets become public. Additionally, if you are using AWS Organizations, you can create a service control policy that prevents changes to Amazon S3 public access policies. AWS Trusted Advisor checks for Amazon S3 buckets that have open access permissions. Bucket permissions that grant, upload, or delete access to everyone create potential security issues by allowing anyone to add, modify, or remove items in a bucket. The Trusted Advisor check examines explicit bucket permissions and associated bucket policies that might override the bucket permissions. You also can use AWS Config to monitor your Amazon S3 buckets for public access. For more information, see How to Use AWS Config to Monitor for and Respond to Amazon S3 Buckets Allowing Public Access.

When reviewing access controls for Amazon S3 buckets, it is important to consider the nature of the data stored within them. Amazon Macie is a service designed to help you discover and protect sensitive data, such as Personally Identifiable Information (PII), Protected Health Information (PHI), and credentials like private keys or AWS access keys.

Resources

Related documents:

- Using AWS Identity and Access Management Access Analyzer
- AWS Control Tower controls library
- AWS Foundational Security Best Practices standard
- AWS Config Managed Rules
- AWS Trusted Advisor check reference
- Monitoring AWS Trusted Advisor check results with Amazon EventBridge
- Managing AWS Config Rules Across All Accounts in Your Organization
- AWS Config and AWS Organizations

- Make your AMI publicly available for use in Amazon EC2

Related videos:

- Best Practices for securing your multi-account environment
- Dive Deep into IAM Access Analyzer

SEC03-BP08 Share resources securely within your organization

As the number of workloads grows, you might need to share access to resources in those workloads or provision the resources multiple times across multiple accounts. You might have constructs to compartmentalize your environment, such as having development, testing, and production environments. However, having separation constructs does not limit you from being able to share securely. By sharing components that overlap, you can reduce operational overhead and allow for a consistent experience without guessing what you might have missed while creating the same resource multiple times.

Desired outcome: Minimize unintended access by using secure methods to share resources within your organization, and help with your data loss prevention initiative. Reduce your operational overhead compared to managing individual components, reduce errors from manually creating the same component multiple times, and increase your workloads' scalability. You can benefit from decreased time to resolution in multi-point failure scenarios, and increase your confidence in determining when a component is no longer needed. For prescriptive guidance on analyzing externally shared resources, see SEC03-BP07 Analyze public and cross-account access.

Common anti-patterns:

- Lack of process to continually monitor and automatically alert on unexpected external share.
- Lack of baseline on what should be shared and what should not.
- Defaulting to a broadly open policy rather than sharing explicitly when required.
- Manually creating foundational resources that overlap when required.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Architect your access controls and patterns to govern the consumption of shared resources securely and only with trusted entities. Monitor shared resources and review shared resource

access continuously, and be alerted on inappropriate or unexpected sharing. Review Analyze public and cross-account access to help you establish governance to reduce the external access to only resources that require it, and to establish a process to monitor continuously and alert automatically.

Cross-account sharing within AWS Organizations is supported by a number of AWS services, such as AWS Security Hub CSPM, Amazon GuardDuty, and AWS Backup. These services allow for data to be shared to a central account, be accessible from a central account, or manage resources and data from a central account. For example, AWS Security Hub CSPM can transfer findings from individual accounts to a central account where you can view all the findings. AWS Backup can take a backup for a resource and share it across accounts. You can use AWS Resource Access Manager (AWS RAM) to share other common resources, such as VPC subnets and Transit Gateway attachments, AWS Network Firewall, or Amazon SageMaker AI pipelines.

To restrict your account to only share resources within your organization, use service control policies (SCPs) to prevent access to external principals. When sharing resources, combine identity-based controls and network controls to create a data perimeter for your organization to help protect against unintended access. A data perimeter is a set of preventive guardrails to help verify that only your trusted identities are accessing trusted resources from expected networks. These controls place appropriate limits on what resources can be shared and prevent sharing or exposing resources that should not be allowed. For example, as a part of your data perimeter, you can use VPC endpoint policies and the `AWS:PrincipalOrgId` condition to ensure the identities accessing your Amazon S3 buckets belong to your organization. It is important to note that SCPs do not apply to service-linked roles or AWS service principals.

When using Amazon S3, turn off ACLs for your Amazon S3 bucket and use IAM policies to define access control. For restricting access to an Amazon S3 origin from Amazon CloudFront, migrate from origin access identity (OAI) to origin access control (OAC) which supports additional features including server-side encryption with AWS Key Management Service.

In some cases, you might want to allow sharing resources outside of your organization or grant a third party access to your resources. For prescriptive guidance on managing permissions to share resources externally, see Permissions management.

Implementation steps

1. **Use AWS Organizations:** AWS Organizations is an account management service that allows you to consolidate multiple AWS accounts into an organization that you create and centrally manage. You can group your accounts into organizational units (OUs) and attach different

policies to each OU to help you meet your budgetary, security, and compliance needs. You can also control how AWS artificial intelligence (AI) and machine learning (ML) services can collect and store data, and use the multi-account management of the AWS services integrated with Organizations.

2. **Integrate AWS Organizations with AWS services:** When you use an AWS service to perform tasks on your behalf in the member accounts of your organization, AWS Organizations creates an IAM service-linked role (SLR) for that service in each member account. You should manage trusted access using the AWS Management Console, the AWS APIs, or the AWS CLI. For prescriptive guidance on turning on trusted access, see Using AWS Organizations with other AWS services and AWS services that you can use with Organizations.
3. **Establish a data perimeter:** A data perimeter provides a clear boundary of trust and ownership. On AWS, it is typically represented as your AWS organization managed by AWS Organizations, along with any on-premises networks or systems that access your AWS resources. The goal of the data perimeter is to verify that access is allowed if the identity is trusted, the resource is trusted, and the network is expected. However, establishing a data perimeter is not a one-size-fits-all approach. Evaluate and adopt the control objectives outlined in the Building a Perimeter on AWS whitepaper based on your specific security risk models and requirements. You should carefully consider your unique risk posture and implement the perimeter controls that align with your security needs.
4. **Use resource sharing in AWS services and restrict accordingly:** Many AWS services allow you to share resources with another account, or target a resource in another account, such as Amazon Machine Images (AMIs) and AWS Resource Access Manager (AWS RAM). Restrict the `ModifyImageAttribute` API to specify the trusted accounts to share the AMI with. Specify the `ram:RequestedAllowsExternalPrincipals` condition when using AWS RAM to constrain sharing to your organization only, to help prevent access from untrusted identities. For prescriptive guidance and considerations, see Resource sharing and external targets.
5. **Use AWS RAM to share securely in an account or with other AWS accounts:** AWS RAM helps you securely share the resources that you have created with roles and users in your account and with other AWS accounts. In a multi-account environment, AWS RAM allows you to create a resource once and share it with other accounts. This approach helps reduce your operational overhead while providing consistency, visibility, and auditability through integrations with Amazon CloudWatch and AWS CloudTrail, which you do not receive when using cross-account access.

If you have resources that you shared previously using a resource-based policy, you can use the `PromoteResourceShareCreatedFromPolicy` API or an equivalent to promote the resource share to a full AWS RAM resource share.

In some cases, you might need to take additional steps to share resources. For example, to share an encrypted snapshot, you need to share a AWS KMS key.

Resources

Related best practices:

- SEC03-BP07 Analyze public and cross-account access
- SEC03-BP09 Share resources securely with a third party
- SEC05-BP01 Create network layers

Related documents:

- Bucket owner granting cross-account permission to objects it does not own
- How to use Trust Policies with IAM
- Building Data Perimeter on AWS
- How to use an external ID when granting a third party access to your AWS resources
- AWS services you can use with AWS Organizations
- Establishing a data perimeter on AWS: Allow only trusted identities to access company data

Related videos:

- Granular Access with AWS Resource Access Manager
- Securing your data perimeter with VPC endpoints
- Establishing a data perimeter on AWS

Related tools:

- Data Perimeter Policy Examples

SEC03-BP09 Share resources securely with a third party

The security of your cloud environment doesn't stop at your organization. Your organization might rely on a third party to manage a portion of your data. The permission management for the third-party managed system should follow the practice of just-in-time access using the principle of least privilege with temporary credentials. By working closely with a third party, you can reduce the scope of impact and risk of unintended access together.

Desired outcome: You avoid using long-term AWS Identity and Access Management (IAM) credentials like access keys and secret keys, as they pose a security risk if misused. Instead, you use IAM roles and temporary credentials to improve your security posture and minimize the operational overhead of managing long-term credentials. When granting third-party access, you use a universally unique identifier (UUID) as the external ID in the IAM trust policy and keep the IAM policies attached to the role under your control to ensure least privilege access. For prescriptive guidance on analyzing externally shared resources, see SEC03-BP07 Analyze public and cross-account access.

Common anti-patterns:

- Using the default IAM trust policy without any conditions.
- Using long-term IAM credentials and access keys.
- Reusing external IDs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

You might want to allow sharing resources outside of AWS Organizations or grant a third party access to your account. For example, a third party might provide a monitoring solution that needs to access resources within your account. In those cases, create an IAM cross-account role with only the privileges needed by the third party. Additionally, define a trust policy using the external ID condition. When using an external ID, you or the third party can generate a unique ID for each customer, third party, or tenancy. The unique ID should not be controlled by anyone but you after it's created. The third party must implement a process to relate the external ID to the customer in a secure, auditable, and reproduceable manner.

You can also use IAM Roles Anywhere to manage IAM roles for applications outside of AWS that use AWS APIs.

If the third party no longer requires access to your environment, remove the role. Avoid providing long-term credentials to a third party. Maintain awareness of other AWS services that support sharing, such as the AWS Well-Architected Tool allowing sharing a workload with other AWS accounts, and AWS Resource Access Manager helping you securely share an AWS resource you own with other accounts.

Implementation steps

1. **Use cross-account roles to provide access to external accounts.** Cross-account roles reduce the amount of sensitive information that is stored by external accounts and third parties for servicing their customers. Cross-account roles allow you to grant access to AWS resources in your account securely to a third party, such as AWS Partners or other accounts in your organization, while maintaining the ability to manage and audit that access. The third party might be providing service to you from a hybrid infrastructure or alternatively pulling data into an offsite location. IAM Roles Anywhere helps you allow third-party workloads to securely interact with your AWS workloads and further reduce the need for long-term credentials.

   You should not use long-term credentials or access keys associated with users to provide external account access. Instead, use cross-account roles to provide the cross-account access.
2. **Perform due diligence and ensure secure access for third-party SaaS providers.** When sharing resources with third-party SaaS providers, perform thorough due diligence to ensure they have a secure and responsible approach to accessing your AWS resources. Evaluate their shared responsibility model to understand what security measures they provide and what falls under your responsibility. Ensure that the SaaS provider has a secure and auditable process for accessing your resources, including the use of external IDs and least privilege access principles. The use of external IDs helps address the confused deputy problem.

   Implement security controls to ensure secure access and adherence to the principle of least privilege when granting access to third-party SaaS providers. This may include the use of external IDs, universally unique identifiers (UUIDs), and IAM trust policies that limit access to only what is strictly necessary. Work closely with the SaaS provider to establish secure access mechanisms, regularly review their access to your AWS resources, and conduct audits to ensure compliance with your security requirements.
3. **Deprecate customer-provided long-term credentials.** Deprecate the use of long-term credentials and use cross-account roles or IAM Roles Anywhere. If you must use long-term credentials, establish a plan to migrate to role-based access. For details on managing keys, see Identity management. Also, work with your AWS account team and the third party to establish a

risk mitigation runbook. For prescriptive guidance on responding to and mitigating the potential impact of a security incident, see Incident response.

4. **Verify that setup has prescriptive guidance or is automated.** The external ID is not treated as a secret, but the external ID must not be an easily guessable value, such as a phone number, name, or account ID. Make the external ID a read-only field so that the external ID cannot be changed for the purpose of impersonating the setup.

   You or the third party can generate the external ID. Define a process to determine who is responsible for generating the ID. Regardless of the entity creating the external ID, the third party enforces uniqueness and formats consistently across customers.

   The policy created for cross-account access in your accounts must follow the least-privilege principle. The third party must provide a role policy document or automated setup mechanism that uses an AWS CloudFormation template or an equivalent for you. This reduces the chance of errors associated with manual policy creation and offers an auditable trail. For more information on using an AWS CloudFormation template to create cross-account roles, see Cross-Account Roles.

   The third party should provide an automated, auditable setup mechanism. However, by using the role policy document outlining the access needed, you should automate the setup of the role. Using an AWS CloudFormation template or equivalent, you should monitor for changes with drift detection as part of the audit practice.

5. **Account for changes.** Your account structure, your need for the third party, or their service offering being provided might change. You should anticipate changes and failures, and plan accordingly with the right people, process, and technology. Audit the level of access you provide on a periodic basis, and implement detection methods to alert you to unexpected changes. Monitor and audit the use of the role and the datastore of the external IDs. You should be prepared to revoke third-party access, either temporarily or permanently, as a result of unexpected changes or access patterns. Also, measure the impact to your revocation operation, including the time it takes to perform, the people involved, the cost, and the impact to other resources.

   For prescriptive guidance on detection methods, see the Detection best practices.

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC03-BP05 Define permission guardrails for your organization
- SEC03-BP06 Manage access based on lifecycle
- SEC03-BP07 Analyze public and cross-account access
- SEC04 Detection

Related documents:

- Bucket owner granting cross-account permission to objects it does not own
- How to use trust policies with IAM roles
- Delegate access across AWS accounts using IAM roles
- How do I access resources in another AWS account using IAM?
- Security best practices in IAM
- Cross-account policy evaluation logic
- How to use an external ID when granting access to your AWS resources to a third party
- Collecting Information from AWS CloudFormation Resources Created in External Accounts with Custom Resources
- Securely Using External ID for Accessing AWS Accounts Owned by Others
- Extend IAM roles to workloads outside of IAM with IAM Roles Anywhere

Related videos:

- How do I allow users or roles in a separate AWS account access to my AWS account?
- AWS re:Invent 2018: Become an IAM Policy Master in 60 Minutes or Less
- AWS Knowledge Center Live: IAM Best Practices and Design Decisions

Related examples:

- Configure cross-account access to Amazon DynamoDB
- AWS STS Network Query Tool

Detection

Question

- SEC 4. How do you detect and investigate security events?

SEC 4. How do you detect and investigate security events?

Capture and analyze events from logs and metrics to gain visibility. Take action on security events and potential threats to help secure your workload.

Best practices

- SEC04-BP01 Configure service and application logging
- SEC04-BP02 Capture logs, findings, and metrics in standardized locations
- SEC04-BP03 Correlate and enrich security alerts
- SEC04-BP04 Initiate remediation for non-compliant resources

SEC04-BP01 Configure service and application logging

Retain security event logs from services and applications. This is a fundamental principle of security for audit, investigations, and operational use cases, and a common security requirement driven by governance, risk, and compliance (GRC) standards, policies, and procedures.

Desired outcome: An organization should be able to reliably and consistently retrieve security event logs from AWS services and applications in a timely manner when required to fulfill an internal process or obligation, such as a security incident response. Consider centralizing logs for better operational results.

Common anti-patterns:

- Logs are stored in perpetuity or deleted too soon.
- Everybody can access logs.
- Relying entirely on manual processes for log governance and use.
- Storing every single type of log just in case it is needed.
- Checking log integrity only when necessary.

Benefits of establishing this best practice: Implement a root cause analysis (RCA) mechanism for security incidents and a source of evidence for your governance, risk, and compliance obligations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

During a security investigation or other use cases based on your requirements, you need to be able to review relevant logs to record and understand the full scope and timeline of the incident. Logs are also required for alert generation, indicating that certain actions of interest have happened. It is critical to select, turn on, store, and set up querying and retrieval mechanisms and alerting.

Implementation steps

- **Select and use log sources.** Ahead of a security investigation, you need to capture relevant logs to retroactively reconstruct activity in an AWS account. Select log sources relevant to your workloads.

  The log source selection criteria should be based on the use cases required by your business. Establish a trail for each AWS account using AWS CloudTrail or an AWS Organizations trail, and configure an Amazon S3 bucket for it.

  AWS CloudTrail is a logging service that tracks API calls made against an AWS account capturing AWS service activity. It's turned on by default with a 90-day retention of management events that can be retrieved through CloudTrail Event history using the AWS Management Console, the AWS CLI, or an AWS SDK. For longer retention and visibility of data events, create a CloudTrail trail and associate it with an Amazon S3 bucket, and optionally with a Amazon CloudWatch log group. Alternatively, you can create a CloudTrail Lake, which retains CloudTrail logs for up to seven years and provides a SQL-based querying facility

  AWS recommends that customers using a VPC turn on network traffic and DNS logs using VPC Flow Logs and Amazon Route 53 resolver query logs, respectively, and streaming them to either an Amazon S3 bucket or a CloudWatch log group. You can create a VPC flow log for a VPC, a subnet, or a network interface. For VPC Flow Logs, you can be selective on how and where you use Flow Logs to reduce cost.

  AWS CloudTrail Logs, VPC Flow Logs, and Route 53 resolver query logs are the basic logging sources to support security investigations in AWS. You can also use Amazon Security Lake to collect, normalize, and store this log data in Apache Parquet format and Open Cybersecurity

Schema Framework (OCSF), which is ready for querying. Security Lake also supports other AWS logs and logs from third-party sources.

AWS services can generate logs not captured by the basic log sources, such as Elastic Load Balancing logs, AWS WAF logs, AWS Config recorder logs, Amazon GuardDuty findings, Amazon Elastic Kubernetes Service (Amazon EKS) audit logs, and Amazon EC2 instance operating system and application logs. For a full list of logging and monitoring options, see Appendix A: Cloud capability definitions – Logging and Events of the AWS Security Incident Response Guide.

- **Research logging capabilities for each AWS service and application:** Each AWS service and application provides you with options for log storage, each of which with its own retention and life-cycle capabilities. The two most common log storage services are Amazon Simple Storage Service (Amazon S3) and Amazon CloudWatch. For long retention periods, it is recommended to use Amazon S3 for its cost effectiveness and flexible lifecycle capabilities. If the primary logging option is Amazon CloudWatch Logs, as an option, you should consider archiving less frequently accessed logs to Amazon S3.
- **Select log storage:** The choice of log storage is generally related to which querying tool you use, retention capabilities, familiarity, and cost. The main options for log storage are an Amazon S3 bucket or a CloudWatch Log group.

  An Amazon S3 bucket provides cost-effective, durable storage with an optional lifecycle policy. Logs stored in Amazon S3 buckets can be queried using services such as Amazon Athena.

  A CloudWatch log group provides durable storage and a built-in query facility through CloudWatch Logs Insights.
- **Identify appropriate log retention:** When you use an Amazon S3 bucket or CloudWatch log group to store logs, you must establish adequate lifecycles for each log source to optimize storage and retrieval costs. Customers generally have between three months to one year of logs readily available for querying, with retention of up to seven years. The choice of availability and retention should align with your security requirements and a composite of statutory, regulatory, and business mandates.
- **Use logging for each AWS service and application with proper retention and lifecycle policies:** For each AWS service or application in your organization, look for the specific logging configuration guidance:
  - Configure AWS CloudTrail Trail
  - Configure VPC Flow Logs
  - Configure Amazon GuardDuty Finding Export

  - Configure AWS Config recording
  - Configure AWS WAF web ACL traffic
  - Configure AWS Network Firewall network traffic logs
  - Configure Elastic Load Balancing access logs
  - Configure Amazon Route 53 resolver query logs
  - Configure Amazon RDS logs
  - Configure Amazon EKS Control Plane logs
  - Configure Amazon CloudWatch agent for Amazon EC2 instances and on-premises servers
- **Select and implement querying mechanisms for logs:** For log queries, you can use CloudWatch Logs Insights for data stored in CloudWatch log groups, and Amazon Athena and Amazon OpenSearch Service for data stored in Amazon S3. You can also use third-party querying tools such as a security information and event management (SIEM) service.

  The process for selecting a log querying tool should consider the people, process, and technology aspects of your security operations. Select a tool that fulfills operational, business, and security requirements, and is both accessible and maintainable in the long term. Keep in mind that log querying tools work optimally when the number of logs to be scanned is kept within the tool's limits. It is not uncommon to have multiple querying tools because of cost or technical constraints.

  For example, you might use a third-party security information and event management (SIEM) tool to perform queries for the last 90 days of data, but use Athena to perform queries beyond 90 days because of the log ingestion cost of a SIEM. Regardless of the implementation, verify that your approach minimizes the number of tools required to maximize operational efficiency, especially during a security event investigation.
- **Use logs for alerting:** AWS provides alerting through several security services:
  - AWS Config monitors and records your AWS resource configurations and allows you to automate the evaluation and remediation against desired configurations.
  - Amazon GuardDuty is a threat detection service that continually monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads. GuardDuty ingests, aggregates, and analyzes information from sources, such as AWS CloudTrail management and data events, DNS logs, VPC Flow Logs, and Amazon EKS Audit logs. GuardDuty pulls independent data streams directly from CloudTrail, VPC Flow Logs, DNS query logs, and Amazon EKS. You don't have to manage Amazon S3 bucket policies or modify

the way you collect and store logs. It is still recommended to retain these logs for your own investigation and compliance purposes.

- AWS Security Hub CSPM provides a single place that aggregates, organizes, and prioritizes your security alerts or findings from multiple AWS services and optional third-party products to give you a comprehensive view of security alerts and compliance status.

You can also use custom alert generation engines for security alerts not covered by these services or for specific alerts relevant to your environment. For information on building these alerts and detections, see Detection in the AWS Security Incident Response Guide.

Resources

Related best practices:

- SEC04-BP02 Capture logs, findings, and metrics in standardized locations
- SEC07-BP04 Define scalable data lifecycle management
- SEC10-BP06 Pre-deploy tools

Related documents:

- AWS Security Incident Response Guide
- Getting Started with Amazon Security Lake
- Getting started: Amazon CloudWatch Logs

Related videos:

- AWS re:Invent 2022 - Introducing Amazon Security Lake

Related examples:

- Assisted Log Enabler for AWS
- AWS Security Hub CSPM Findings Historical Export

SEC04-BP02 Capture logs, findings, and metrics in standardized locations

Security teams rely on logs and findings to analyze events that may indicate unauthorized activity or unintentional changes. To streamline this analysis, capture security logs and findings in standardized locations. This makes data points of interest available for correlation and can simplify tool integrations.

Desired outcome: You have a standardized approach to collect, analyze, and visualize log data, findings, and metrics. Security teams can efficiently correlate, analyze, and visualize security data across disparate systems to discover potential security events and identify anomalies. Security information and event management (SIEM) systems or other mechanisms are integrated to query and analyze log data for timely responses, tracking, and escalation of security events.

Common anti-patterns:

- Teams independently own and manage logging and metrics collection that is inconsistent to the organization's logging strategy.
- Teams don't have adequate access controls to restrict visibility and alteration of the data collected.
- Teams don't govern their security logs, findings, and metrics as part of their data classification policy.
- Teams neglect data sovereignty and localization requirements when configuring data collections.

Benefits of establishing this best practice: A standardized logging solution to collect and query log data and events improves insights derived from the information they contain. Configuring an automated lifecycle for the collected log data can reduce the costs incurred by log storage. You can build fine-grained access control for the collected log information according to the sensitivity of the data and access patterns needed by your teams. You can integrate tooling to correlate, visualize, and derive insights from the data.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Growth in AWS usage within an organization results in a growing number of distributed workloads and environments. As each of these workloads and environments generate data about the activity within them, capturing and storing this data locally presents a challenge for security operations. Security teams use tools such as security information and event management (SIEM) systems to

collect data from distributed sources and undergo correlation, analysis, and response workflows. This requires managing a complex set of permissions for accessing the various data sources and additional overhead in operating the extraction, transformation, and loading (ETL) processes.

To overcome these challenges, consider aggregating all relevant sources of security log data into a Log Archive account as described in Organizing Your AWS Environment Using Multiple Accounts. This includes all security-related data from your workload and logs that AWS services generate, such as AWS CloudTrail, AWS WAF, Elastic Load Balancing, and Amazon Route 53. There are several benefits to capturing this data in standardized locations in a separate AWS account with proper cross-account permissions. This practice helps prevent log tampering within compromised workloads and environments, provides a single integration point for additional tools, and offers a more simplified model for configuring data retention and lifecycle. Evaluate the impacts of data sovereignty, compliance scopes, and other regulations to determine if multiple security data storage locations and retention periods are required.

To ease capturing and standardizing logs and findings, evaluate Amazon Security Lake in your Log Archive account. You can configure Security Lake to automatically ingest data from common sources such as CloudTrail, Route 53, Amazon EKS, and VPC Flow Logs. You can also configure AWS Security Hub CSPM as a data source into Security Lake, allowing you to correlate findings from other AWS services, such as Amazon GuardDuty and Amazon Inspector, with your log data. You can also use third-party data source integrations, or configure custom data sources. All integrations standardize your data into the Open Cybersecurity Schema Framework (OCSF) format, and are stored in Amazon S3 buckets as Parquet files, eliminating the need for ETL processing.

Storing security data in standardized locations provides advanced analytics capabilities. AWS recommends you deploy tools for security analytics that operate in an AWS environment into a Security Tooling account that is separate from your Log Archive account. This approach allows you to implement controls at depth to protect the integrity and availability of the logs and log management process, distinct from the tools that access them. Consider using services, such as Amazon Athena, to run on-demand queries that correlate multiple data sources. You can also integrate visualization tools, such as Quick. AI-powered solutions are becoming increasingly available and can perform functions such as translating findings into human-readable summaries and natural language interaction. These solutions are often more readily integrated by having a standardized data storage location for querying.

Implementation steps

1. **Create the Log Archive and Security Tooling accounts**

   a. Using AWS Organizations, create the Log Archive and Security Tooling accounts under a security organizational unit. If you are using AWS Control Tower to manage your organization, the Log Archive and Security Tooling accounts are created for you automatically. Configure roles and permissions for accessing and administering these accounts as required.

2. **Configure your standardized security data locations**

   a. Determine your strategy for creating standardized security data locations. You can achieve this through options like common data lake architecture approaches, third-party data products, or Amazon Security Lake. AWS recommends that you capture security data from AWS Regions that are opted-in for your accounts, even when not actively in use.

3. **Configure data source publication to your standardized locations**

   a. Identify the sources for your security data and configure them to publish into your standardized locations. Evaluate options to automatically export data in the desired format as opposed to those where ETL processes need to be developed. With Amazon Security Lake, you can collect data from supported AWS sources and integrated third-party systems.

4. **Configure tools to access your standardized locations**

   a. Configure tools such as Amazon Athena, Quick, or third-party solutions to have the access required to your standardized locations. Configure these tools to operate out of the Security Tooling account with cross-account read access to the Log Archive account where applicable. Create subscribers in Amazon Security Lake to provide these tools access to your data.

Resources

Related best practices:

- SEC01-BP01 Separate workloads using accounts
- SEC07-BP04 Define data lifecycle management
- SEC08-BP04 Enforce access control
- OPS08-BP02 Analyze workload logs

Related documents:

- AWS Whitepapers: Organizing Your AWS Environment Using Multiple Accounts
- AWS Prescriptive Guidance: AWS Security Reference Architecture (AWS SRA)
- AWS Prescriptive Guidance: Logging and monitoring guide for application owners

Related examples:

- Aggregating, searching, and visualizing log data from distributed sources with Amazon Athena and Quick
- How to visualize Amazon Security Lake findings with Quick
- Generate AI powered insights for Amazon Security Lake using Amazon SageMaker AI Studio and Amazon Bedrock
- Identify cybersecurity anomalies in your Amazon Security Lake data using Amazon SageMaker AI
- Ingest, transform, and deliver events published by Amazon Security Lake to Amazon OpenSearch Service
- Simplify AWS CloudTrail log analysis with natural language query generation in CloudTrail Lake

Related tools:

- Amazon Security Lake
- Amazon Security Lake Partner Integrations
- Open Cybersecurity Schema Framework (OCSF)
- Amazon Athena
- Quick
- Amazon Bedrock

SEC04-BP03 Correlate and enrich security alerts

Unexpected activity can generate multiple security alerts by different sources, requiring further correlation and enrichment to understand the full context. Implement automated correlation and enrichment of security alerts to help achieve more accurate incident identification and response.

Desired outcome: As activity generates different alerts within your workloads and environments, automated mechanisms correlate data and enrich that data with additional information. This pre-processing presents a more detailed understanding of the event, which helps your investigators determine the criticality of the event and if it constitutes an incident that requires formal response. This process reduces the load on your monitoring and investigation teams.

Common anti-patterns:

- Different groups of people investigate findings and alerts generated by different systems, unless otherwise mandated by separation of duty requirements.
- Your organization funnels all security finding and alert data to standard locations, but requires investigators to perform manual correlation and enrichment.
- You rely solely on the intelligence of threat detection systems to report on findings and establish criticality.

Benefits of establishing this best practice: Automated correlation and enrichment of alerts helps to reduce the overall cognitive load and manual data preparation required of your investigators. This practice can reduce the time it takes to determine if the event represents an incident and initiate a formal response. Additional context also helps you accurately assess the true severity of an event, as it may be higher or lower than what any one alert suggests.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Security alerts can come from many different sources within AWS, including:

- Services such as Amazon GuardDuty, AWS Security Hub CSPM, Amazon Macie, Amazon Inspector, AWS Config, AWS Identity and Access Management Access Analyzer, and Network Access Analyzer
- Alerts from automated analysis of AWS service, infrastructure, and application logs, such as from Security Analytics for Amazon OpenSearch Service.
- Alarms in response to changes in your billing activity from sources such as Amazon CloudWatch, Amazon EventBridge, or AWS Budgets.
- Third-party sources such as threat intelligence feeds and Security Partner Solutions from the AWS Partner Network
- Contact by AWS Trust & Safety or other sources, such as customers or internal employees.
- Use Threat Technique Catalog by AWS (TTC) to assist with identification and correlation of threat actor behavior through indicator of compromise (IoC) identification. The TTC is an extension of the MITRE ATT&CK framework, categorizing all known and observed threat actor behaviors and techniques directed at AWS resources.

In their most fundamental form, alerts contain information about who (the *principal* or *identity*) is doing what *(*the *action* taken) to what (the *resources* affected). For each of these sources, identify

if there are ways you can create mappings across identifiers for these identities, actions, and resources as the foundation for performing correlation. This can take the form of integrating alert sources with a security information and event management (SIEM) tool to perform automated correlation for you, building your own data pipelines and processing, or a combination of both.

An example of a service that can perform correlation for you is Amazon Detective. Detective performs ongoing ingestion of alerts from various AWS and third-party sources and uses different forms of intelligence to assemble a visual graph of their relationships to aid investigations.

While the initial criticality of an alert is an aid for prioritization, the context in which the alert happened determines its true criticality. As an example, Amazon GuardDuty can alert that an Amazon EC2 instance within your workload is querying an unexpected domain name. GuardDuty might assign low criticality to this alert on its own. However, automated correlation with other activity around the time of the alert might uncover that several hundred EC2 instances were deployed by the same identity, which increases overall operating costs. In this event, this correlated event context would warrant a new security alert and the criticality might be adjusted to high, which would expedite further action.

Implementation steps

1. Identify sources for security alert information. Understand how alerts from these systems represent identity, action, and resources to determine where correlation is possible.
2. Establish a mechanism for capturing alerts from different sources. Consider services such as Security Hub CSPM, EventBridge, and CloudWatch for this purpose.
3. Identify sources for data correlation and enrichment. Example sources include AWS CloudTrail, VPC Flow Logs, Route 53 Resolver logs, and infrastructure and application logs. Any or all of these logs might be consumed through a single integration with Amazon Security Lake.
4. Integrate your alerts with your data correlation and enrichment sources to create more detailed security event contexts and establish criticality.
    a. Amazon Detective, SIEM tooling, or other third-party solutions can perform a certain level of ingestion, correlation, and enrichment automatically.
    b. You can also use AWS services to build your own. For example, you can invoke an AWS Lambda function to run an Amazon Athena query against AWS CloudTrail or Amazon Security Lake, and publish the results to EventBridge.

Resources

Related best practices:

- SEC10-BP03 Prepare forensic capabilities
- OPS08-BP04 Create actionable alerts
- REL06-BP03 Send notifications (Real-time processing and alarming)

Related documents:

- AWS Security Incident Response Guide

Related examples:

- How to enrich AWS Security Hub CSPM findings with account metadata

Related tools:

- Amazon Detective
- Amazon EventBridge
- AWS Lambda
- Amazon Athena

SEC04-BP04 Initiate remediation for non-compliant resources

Your detective controls may alert on resources that are out of compliance with your configuration requirements. You can initiate programmatically-defined remediations, either manually or automatically, to fix these resources and help minimize potential impacts. When you define remediations programmatically, you can take prompt and consistent action.

While automation can enhance security operations, you should implement and manage automation carefully. Place appropriate oversight and control mechanisms to verify that automated responses are effective, accurate, and aligned with organizational policies and risk appetite.

Desired outcome: You define resource configuration standards along with the steps to remediate when resources are detected to be non-compliant. Where possible, you've defined remediations

programmatically so they can be initiated either manually or through automation. Detection systems are in place to identify non-compliant resources and publish alerts into centralized tools that are monitored by your security personnel. These tools support running your programmatic remediations, either manually or automatically. Automatic remediations have appropriate oversight and control mechanisms in place to govern their use.

Common anti-patterns:

- You implement automation, but fail to thoroughly test and validate remediation actions. This can result in unintended consequences, such as disrupting legitimate business operations or causing system instability.
- You improve response times and procedures through automation, but without proper monitoring and mechanisms that allow human intervention and judgment when needed.
- You rely solely on remediations, rather than having remediations as one part of a broader incident response and recovery program.

Benefits of establishing this best practice: Automatic remediations can respond to misconfigurations faster than manual processes, which helps you minimize potential business impacts and reduce the window of opportunity for unintended uses. When you define remediations programmatically, they are applied consistently, which reduces the risk of human error. Automation also can handle a larger volume of alerts simultaneously, which is particularly important in environments operating at large scale.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As described in SEC01-BP03 Identify and validate control objectives, services such as AWS Config and AWS Security Hub CSPM can help you monitor the configuration of resources in your accounts for adherence to your requirements. When non-compliant resources are detected, services such as AWS Security Hub CSPM, can help with routing alerts appropriately and remediation. These solutions provide a central place for your security investigators to monitor for issues and take corrective action.

In addition to AWS Security Hub CSPM, AWS introduced Security Hub Advanced. This service, announced at re:Invent 2025, transforms how organizations prioritize their most critical security issues and respond at scale to protect their cloud environments. The enhanced Security Hub now uses advanced analytics to automatically correlate, enrich, and prioritize security signals across

your cloud environment. Security Hub seamlessly integrates with Amazon GuardDuty, Amazon Inspector, Amazon Macie, and AWS Security Hub CSPM. Correlated findings in Security Hub can result in a net-new finding, called an exposure finding, which includes an assumed attack path based on vulnerabilities found in each resource.

While some non-compliant resource situations are unique and require human judgment to remediate, other situations have a standard response that you can define programmatically. For example, a standard response to a misconfigured VPC security group could be to remove the disallowed rules and notify the owner. Responses can be defined in AWS Lambda functions, AWS Systems Manager Automation documents, or through other code environments you prefer. Make sure the environment is able to authenticate to AWS using an IAM role with the least amount of permission needed to take corrective action.

Once you define the desired remediation, you can then determine your preferred means for initiating it. AWS Config can initiate remediations for you. If you are using Security Hub CSPM, you can do this through custom actions, which publishes the finding information to Amazon EventBridge. An EventBridge rule can then initiate your remediation. You can configure remediations through Security Hub CSPM to run either automatically or manually.

For programmatic remediation, we recommend that you have comprehensive logs and audits for the actions taken, as well as their outcomes. Review and analyze these logs to assess the effectiveness of the automated processes, and identify areas of improvement. Capture logs in Amazon CloudWatch Logs and remediation outcomes as finding notes in Security Hub CSPM.

As a starting point, consider Automated Security Response on AWS, which has pre-built remediations for resolving common security misconfigurations.

Implementation steps

1. Analyze and prioritize alerts.
    a. Consolidate security alerts from various AWS services into Security Hub CSPM for centralized visibility, prioritization, and remediation.
2. Develop remediations.
    a. Use services such as Systems Manager and AWS Lambda to run programmatic remediations.
3. Configure how remediations are initiated.
    a. Using Systems Manager, define custom actions that publish findings to EventBridge. Configure these actions to be initiated manually or automatically.

   b. You can also use Amazon Simple Notification Service (SNS) to send notifications and alerts to relevant stakeholders (like security team or incident response teams) for manual intervention or escalation, if required.

4. Review and analyze remediation logs for effectiveness and improvement.

   a. Send log output to CloudWatch Logs. Capture outcomes as finding notes in Security Hub CSPM.

Resources

Related best practices:

- SEC06-BP03 Reduce manual management and interactive access

Related documents:

- AWS Security Incident Response Guide - Detection

Related examples:

- Automated Security Response on AWS
- Monitor EC2 instance key pairs using AWS Config
- Create AWS Config custom rules by using AWS CloudFormation Guard policies
- Automatically remediate unencrypted Amazon RDS DB instances and clusters

Related tools:

- AWS Systems Manager Automation
- Automated Security Response on AWS

Infrastructure protection

Questions

- SEC 5. How do you protect your network resources?
- SEC 6. How do you protect your compute resources?

SEC 5. How do you protect your network resources?

Any workload that has some form of network connectivity, whether it's the internet or a private network, requires multiple layers of defense to help protect from external and internal network-based threats.

Best practices

- SEC05-BP01 Create network layers
- SEC05-BP02 Control traffic flow within your network layers
- SEC05-BP03 Implement inspection-based protection
- SEC05-BP04 Automate network protection

SEC05-BP01 Create network layers

Segment your network topology into different layers based on logical groupings of your workload components according to their data sensitivity and access requirements. Distinguish between components that require inbound access from the internet, such as public web endpoints, and those that only need internal access, such as databases.

Desired outcome: The layers of your network are part of an integral defense-in-depth approach to security that complements the identity authentication and authorization strategy of your workloads. Layers are in place according to data sensitivity and access requirements, with appropriate traffic flow and control mechanisms.

Common anti-patterns:

- You create all resources in a single VPC or subnet.
- You construct your network layers without consideration of data sensitivity requirements, component behaviors, or functionality.
- You use VPCs and subnets as defaults for all network layer considerations, and you don't consider how AWS managed services influence your topology.

Benefits of establishing this best practice: Establishing network layers is the first step in restricting unnecessary pathways through the network, particularly those that lead to critical systems and data. This makes it harder for unauthorized actors to gain access to your network and navigate to additional resources within. Discrete network layers beneficially reduce the scope of

analysis for inspection systems, such as for intrusion detection or malware prevention. This reduces the potential for false positives and unnecessary processing overhead.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When designing a workload architecture, it is common to separate components into different layers based on their responsibility. For example, a web application can have a presentation layer, application layer, and data layer. You can take a similar approach when designing your network topology. Underlying network controls can help enforce your workload's data access requirements. For example, in a three-tier web application architecture, you can store your static presentation layer files on Amazon S3 and serve them from a content delivery network (CDN), such as Amazon CloudFront. The application layer can have public endpoints that an Application Load Balancer (ALB) serves in an Amazon VPC public subnet (similar to a demilitarized zone, or DMZ), with back-end services deployed into private subnets. The data layer, that is hosting resources such as databases and shared file systems, can reside in different private subnets from the resources of your application layer. At each of these layer boundaries (CDN, public subnet, private subnet), you can deploy controls that allow only authorized traffic to traverse across those boundaries.

Similar to modeling network layers based on the functional purpose of your workload's components, also consider the sensitivity of the data being processed. Using the web application example, while all of your workload services may reside within the application layer, different services may process data with different sensitivity levels. In this case, dividing the application layer using multiple private subnets, different VPCs in the same AWS account, or even different VPCs in different AWS accounts for each level of data sensitivity may be appropriate according to your control requirements.

A further consideration for network layers is the behavior consistency of your workload's components. Continuing the example, in the application layer you may have services that accept inputs from end-users or external system integrations that are inherently riskier than the inputs to other services. Examples include file uploads, code scripts to run, email scanning and so on. Placing these services in their own network layer helps create a stronger isolation boundary around them, and can prevent their unique behavior from creating false positive alerts in inspection systems.

As part of your design, consider how using AWS managed services influences your network topology. Explore how services such as Amazon VPC Lattice can help make the interoperability of your workload components across network layers easier. When using AWS Lambda, deploy in your

VPC subnets unless there are specific reasons not to. Determine where VPC endpoints and AWS PrivateLink can simplify adhering to security policies that limit access to internet gateways.

Implementation steps

1. Review your workload architecture. Logically group components and services based on the functions they serve, the sensitivity of data being processed, and their behavior.
2. For components responding to requests from the internet, consider using load balancers or other proxies to provide public endpoints. Explore shifting security controls by using managed services, such as CloudFront, Amazon API Gateway, Elastic Load Balancing, and AWS Amplify to host public endpoints.
3. For components running in compute environments, such as Amazon EC2 instances, AWS Fargate containers, or Lambda functions, deploy these into private subnets based on your groups from the first step.
4. For fully managed AWS services, such as Amazon DynamoDB, Amazon Kinesis, or Amazon SQS, consider using VPC endpoints as the default for access over private IP addresses.

Resources

Related best practices:

- REL02 Plan your network topology
- PERF04-BP01 Understand how networking impacts performance

Related videos:

- AWS re:Invent 2023 - AWS networking foundations

Related examples:

- VPC examples
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront

SEC05-BP02 Control traffic flow within your network layers

Within the layers of your network, use further segmentation to restrict traffic only to the flows necessary for each workload. First, focus on controlling traffic between the internet or other external systems to a workload and your environment (*north-south* traffic). Afterwards, look at flows between different components and systems (*east-west* traffic).

Desired outcome: You permit only the network flows necessary for the components of your workloads to communicate with each other and their clients and any other services they depend on. Your design factors in considerations such as public compared to private ingress and egress, data classification, regional regulations, and protocol requirements. Wherever possible, you favor point-to-point flows over network peering as part of a *principle of least privilege* design.

Common anti-patterns:

- You take a perimeter-based approach to network security and only control traffic flow at the boundary of your network layers.
- You assume all traffic within a network layer is authenticated and authorized.
- You apply controls for either your ingress traffic or your egress traffic, but not both.
- You rely solely on your workload components and network controls to authenticate and authorize traffic.

Benefits of establishing this best practice: This practice helps reduce the risk of unauthorized movement within your network and adds an extra layer of authorization to your workloads. By performing traffic flow control, you can restrict the scope of impact of a security incident and speed up detection and response.

Level of risk exposed if this best practice is not established: High

Implementation guidance

While network layers help establish the boundaries around components of your workload that serve a similar function, data sensitivity level, and behavior, you can create a much finer-grained level of traffic control by using techniques to further segment components within these layers that follows the principle of least privilege. Within AWS, network layers are primarily defined using subnets according to IP address ranges within an Amazon VPC. Layers can also be defined using different VPCs, such as for grouping microservice environments by business domain. When using multiple VPCs, mediate routing using an AWS Transit Gateway. While this provides traffic control at a Layer 4 level (IP address and port ranges) using security groups and route tables, you can gain

further control using additional services, such as AWS PrivateLink, Amazon Route 53 Resolver DNS Firewall, AWS Network Firewall, and AWS WAF.

Understand and inventory the data flow and communication requirements of your workloads in terms of connection-initiating parties, ports, protocols, and network layers. Evaluate the protocols available for establishing connections and transmitting data to select ones that achieve your protection requirements (for example, HTTPS rather than HTTP). Capture these requirements at both the boundaries of your networks and within each layer. Once these requirements are identified, explore options to only allow the required traffic to flow at each connection point. A good starting point is to use *security groups* within your VPC, as they can be attached to resources that uses an Elastic Network Interface (ENI), such Amazon EC2 instances, Amazon ECS tasks, Amazon EKS pods, or Amazon RDS databases. Unlike a Layer 4 firewall, a security group can have a rule that allows traffic from another security group by its identifier, minimizing updates as resources within the group change over time. You can also filter traffic using both inbound and outbound rules using security groups.

When traffic moves between VPCs, it's common to use VPC peering for simple routing or the AWS Transit Gateway for complex routing. With these approaches, you facilitate traffic flows between the range of IP addresses of both the source and destination networks. However, if your workload only requires traffic flows between specific components in different VPCs, consider using a point-to-point connection using AWS PrivateLink. To do this, identify which service should act as the producer and which should act as the consumer. Deploy a compatible load balancer for the producer, turn on PrivateLink accordingly, and then accept a connection request by the consumer. The producer service is then assigned a private IP address from the consumer's VPC that the consumer can use to make subsequent requests. This approach reduces the need to peer the networks. Include the costs for data processing and load balancing as part of evaluating PrivateLink.

While security groups and PrivateLink help control the flow between the components of your workloads, another major consideration is how to control which DNS domains your resources are allowed to access (if any). Depending on the DHCP configuration of your VPCs, you can consider two different AWS services for this purpose. Most customers use the default Route 53 Resolver DNS service (also called Amazon DNS server or AmazonProvidedDNS) available to VPCs at the +2 address of its CIDR range. With this approach, you can create DNS Firewall rules and associate them to your VPC that determine what actions to take for the domain lists you supply.

If you are not using the Route 53 Resolver, or if you want to complement the Resolver with deeper inspection and flow control capabilities beyond domain filtering, consider deploying an AWS

Network Firewall. This service inspects individual packets using either stateless or stateful rules to determine whether to deny or allow the traffic. You can take a similar approach for filtering inbound web traffic to your public endpoints using AWS WAF. For further guidance on these services, see SEC05-BP03 Implement inspection-based protection.

Implementation steps

1. Identify the required data flows between the components of your workloads.
2. Apply multiple controls with a defense-in-depth approach for both inbound and outbound traffic, including the use of security groups, and route tables.
3. Use firewalls to define fine-grained control over network traffic in, out, and across your VPCs, such as the Route 53 Resolver DNS Firewall, AWS Network Firewall, and AWS WAF. Consider using the AWS Firewall Manager for centrally configuring and managing your firewall rules across your organization.

Resources

Related best practices:

- REL03-BP01 Choose how to segment your workload
- SEC09-BP02 Enforce encryption in transit

Related documents:

- Security best practices for your VPC
- AWS Network Optimization Tips
- Guidance for Network Security on AWS
- Secure your VPC's outbound network traffic in the AWS Cloud

Related tools:

- AWS Firewall Manager

Related videos:

- AWS Transit Gateway reference architectures for many VPCs

- Application Acceleration and Protection with Amazon CloudFront, AWS WAF, and AWS Shield
- AWS re:Inforce 2023: Firewalls and where to put them

SEC05-BP03 Implement inspection-based protection

Set up traffic inspection points between your network layers to make sure data in transit matches the expected categories and patterns. Analyze traffic flows, metadata, and patterns to help identify, detect, and respond to events more effectively.

Desired outcome: Traffic that traverses between your network layers are inspected and authorized. Allow and deny decisions are based on explicit rules, threat intelligence, and deviations from baseline behaviors. Protections become stricter as traffic moves closer to sensitive data.

Common anti-patterns:

- Relying solely on firewall rules based on ports and protocols. Not taking advantage of intelligent systems.
- Authoring firewall rules based on specific current threat patterns that are subject to change.
- Only inspecting traffic where traffic transits from private to public subnets, or from public subnets to the Internet.
- Not having a baseline view of your network traffic to compare for behavior anomalies.

Benefits of establishing this best practice: Inspection systems allow you to author intelligent rules, such as allowing or denying traffic only when certain conditions within the traffic data exist. Benefit from managed rule sets from AWS and partners, based on the latest threat intelligence, as the threat landscape changes over time. This reduces the overhead of maintaining rules and researching indicators of compromise, reducing the potential for false positives.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Have fine-grained control over both your stateful and stateless network traffic using AWS Network Firewall, or other Firewalls and Intrusion Prevention Systems (IPS) on AWS Marketplace that you can deploy behind a Gateway Load Balancer (GWLB). AWS Network Firewall supports Suricata-compatible open source IPS specifications to help protect your workload.

Both the AWS Network Firewall and vendor solutions that use a GWLB support different inline inspection deployment models. For example, you can perform inspection on a per-VPC basis,

centralize in an inspection VPC, or deploy in a hybrid model where east-west traffic flows through an inspection VPC and Internet ingress is inspected per-VPC. Another consideration is whether the solution supports unwrapping Transport Layer Security (TLS), enabling deep packet inspection for traffic flows initiated in either direction. For more information and in-depth details on these configurations, see the AWS Network Firewall Best Practice guide.

If you are using solutions that perform out-of-band inspections, such as pcap analysis of packet data from network interfaces operating in promiscuous mode, you can configure VPC traffic mirroring. Mirrored traffic counts towards the available bandwidth of your interfaces and is subject to the same data transfer charges as non-mirrored traffic. You can see if virtual versions of these appliances are available on the AWS Marketplace, which may support inline deployment behind a GWLB.

For components that transact over HTTP-based protocols, protect your application from common threats with a web application firewall (WAF). AWS WAF is a web application firewall that lets you monitor and block HTTP(S) requests that match your configurable rules before sending to Amazon API Gateway, Amazon CloudFront, AWS AppSync or an Application Load Balancer. Consider deep packet inspection when you evaluate the deployment of your web application firewall, as some require you to terminate TLS before traffic inspection. To get started with AWS WAF, you can use AWS Managed Rules in combination with your own, or use existing partner integrations.

You can centrally manage AWS WAF, AWS Shield Advanced, AWS Network Firewall, and Amazon VPC security groups across your AWS Organization with AWS Firewall Manager.

Implementation steps

1. Determine if you can scope inspection rules broadly, such as through an inspection VPC, or if you require a more granular per-VPC approach.
2. For inline inspection solutions:
    a. If using AWS Network Firewall, create rules, firewall policies, and the firewall itself. Once these have been configured, you can route traffic to the firewall endpoint to enable inspection.
    b. If using a third-party appliance with a Gateway Load Balancer (GWLB), deploy and configure your appliance in one or more availability zones. Then, create your GWLB, the endpoint service, endpoint, and configure routing for your traffic.
3. For out-of-band inspection solutions:
    1. Turn on VPC Traffic Mirroring on interfaces where inbound and outbound traffic should be mirrored. You can use Amazon EventBridge rules to invoke an AWS Lambda function to turn

on traffic mirroring on interfaces when new resources are created. Point the traffic mirroring sessions to the Network Load Balancer in front of your appliance that processes traffic.

4. For inbound web traffic solutions:
   a. To configure AWS WAF, start by configuring a web access control list (web ACL). The web ACL is a collection of rules with a serially processed default action (ALLOW or DENY) that defines how your WAF handles traffic. You can create your own rules and groups or use AWS managed rule groups in your web ACL.
   b. Once your web ACL is configured, associate the web ACL with an AWS resource (like an Application Load Balancer, API Gateway REST API, or CloudFront distribution) to begin protecting web traffic.

Resources

Related documents:

- What is Traffic Mirroring?
- Implementing inline traffic inspection using third-party security appliances
- AWS Network Firewall example architectures with routing
- Centralized inspection architecture with AWS Gateway Load Balancer and AWS Transit Gateway

Related examples:

- Best practices for deploying Gateway Load Balancer
- TLS inspection configuration for encrypted egress traffic and AWS Network Firewall

Related tools:

- AWS Marketplace IDS/IPS

SEC05-BP04 Automate network protection

Automate the deployment of your network protections using DevOps practices, such as *infrastructure as code* (IaC) and CI/CD pipelines. These practices can help you track changes in your network protections through a version control system, reduce the time it takes to deploy changes, and help detect if your network protections drift from your desired configuration.

Desired outcome: You define network protections with templates and commit them into a version control system. Automated pipelines are initiated when new changes are made that orchestrates their testing and deployment. Policy checks and other static tests are in place to validate changes before deployment. You deploy changes into a staging environment to validate the controls are operating as expected. Deployment into your production environments is also performed automatically once controls are approved.

Common anti-patterns:

- Relying on individual workload teams to each define their complete network stack, protections, and automations. Not publishing standard aspects of the network stack and protections centrally for workload teams to consume.
- Relying on a central network team to define all aspects of the network, protections, and automations. Not delegating workload-specific aspects of the network stack and protections to that workload's team.
- Striking the right balance between centralization and delegation between a network team and workload teams, but not applying consistent testing and deployment standards across your IaC templates and CI/CD pipelines. Not capturing required configurations in tooling that checks your templates for adherence.

Benefits of establishing this best practice: Using templates to define your network protections allows you to track and compare changes over time with a version control system. Using automation to test and deploy changes creates standardization and predictability, increasing the chances of a successful deployment and reducing repetitive manual configurations.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

A number of network protection controls described in SEC05-BP02 Control traffic flows within your network layers and SEC05-BP03 Implement inspection-based protection come with managed rules systems that can update automatically based on the latest threat intelligence. Examples of protecting your web endpoints include AWS WAF managed rules and AWS Shield Advanced automatic application layer DDoS mitigation. Use AWS Network Firewall managed rule groups to stay up to date with low-reputation domain lists and threat signatures as well.

Beyond managed rules, we recommend you use DevOps practices to automate deploying your network resources, protections, and the rules you specify. You can capture these definitions in

AWS CloudFormation or another *infrastructure as code* (IaC) tool of your choice, commit them to a version control system, and deploy them using CI/CD pipelines. Use this approach to gain the traditional benefits of DevOps for managing your network controls, such as more predictable releases, automated testing using tools like AWS CloudFormation Guard, and detecting drift between your deployed environment and your desired configuration.

Based on the decisions you made as part of SEC05-BP01 Create network layers, you may have a central management approach to creating VPCs that are dedicated for ingress, egress, and inspection flows. As described in the AWS Security Reference Architecture (AWS SRA), you can define these VPCs in a dedicated Network infrastructure account. You can use similar techniques to centrally define the VPCs used by your workloads in other accounts, their security groups, AWS Network Firewall deployments, Route 53 Resolver rules and DNS Firewall configurations, and other network resources. You can share these resources with your other accounts with the AWS Resource Access Manager. With this approach, you can simplify the automated testing and deployment of your network controls to the Network account, with only one destination to manage. You can do this in a hybrid model, where you deploy and share certain controls centrally and delegate other controls to the individual workload teams and their respective accounts.

Implementation steps

1. Establish ownership over which aspects of the network and protections are defined centrally, and which your workload teams can maintain.
2. Create environments to test and deploy changes to your network and its protections. For example, use a Network Testing account and a Network Production account.
3. Determine how you will store and maintain your templates in a version control system. Store central templates in a repository that is distinct from workload repositories, while workload templates can be stored in repositories specific to that workload.
4. Create CI/CD pipelines to test and deploy templates. Define tests to check for misconfigurations and that templates adhere to your company standards.

Resources

Related best practices:

- SEC01-BP06 Automate deployment of standard security controls

Related documents:

- AWS Security Reference Architecture - Network account

Related examples:

- AWS Deployment Pipeline Reference Architecture
- NetDevSecOps to modernize AWS networking deployments
- Integrating AWS CloudFormation security tests with AWS Security Hub CSPM and AWS CodeBuild reports

Related tools:

- AWS CloudFormation
- AWS CloudFormation Guard
- cfn_nag

SEC 6. How do you protect your compute resources?

Compute resources in your workload require multiple layers of defense to help protect from external and internal threats. Compute resources include EC2 instances, containers, AWS Lambda functions, database services, IoT devices, and more.

Best practices

- SEC06-BP01 Perform vulnerability management
- SEC06-BP02 Provision compute from hardened images
- SEC06-BP03 Reduce manual management and interactive access
- SEC06-BP04 Validate software integrity
- SEC06-BP05 Automate compute protection

SEC06-BP01 Perform vulnerability management

Frequently scan and patch for vulnerabilities in your code, dependencies, and in your infrastructure to help protect against new threats.

Desired outcome: You have a solution that continually scans your workload for software vulnerabilities, potential defects, and unintended network exposure. You have established processes and procedures to identify, prioritize, and remediate these vulnerabilities based on risk

assessment criteria. Additionally, you have implemented automated patch management for your compute instances. Your vulnerability management program is integrated into your software development lifecycle, with solutions to scan your source code during the CI/CD pipeline.

Common anti-patterns:

- Not having a vulnerability management program.
- Performing system patching without considering severity or risk avoidance.
- Using software that has passed its vendor-provided end of life (EOL) date.
- Deploying code into production before analyzing it for security issues.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Vulnerability management is a key aspect of maintaining a secure and robust cloud environment. It involves a comprehensive process that includes security scans, identification and prioritization of issues, and patch operations to resolve the identified vulnerabilities. Automation plays a pivotal role in this process because it facilitates continuous scanning of workloads for potential issues and unintended network exposure, as well as remediation efforts.

The AWS Shared Responsibility Model is a fundamental concept that underpins vulnerability management. According to this model, AWS is responsible for securing the underlying infrastructure, including hardware, software, networking, and facilities that run AWS services. Conversely, you are responsible for securing your data, security configurations, and management tasks associated with services like Amazon EC2 instances and Amazon S3 objects.

AWS offers a range of services to support vulnerability management programs. Amazon Inspector continuously scans AWS workloads for software vulnerabilities and unintended network access, while AWS Systems Manager Patch Manager helps manage patching across Amazon EC2 instances. These services can be integrated with AWS Security Hub CSPM, a cloud security posture management service that automates AWS security checks, centralizes security alerts, and provides a comprehensive view of an organization's security posture. Furthermore, Amazon CodeGuru Security uses static code analysis to identify potential issues in Java and Python applications during the development phase.

By incorporating vulnerability management practices into the software development lifecycle, you can proactively address vulnerabilities before they are introduced into production environments, which reduces the risk of security events and minimizes the potential impact of vulnerabilities.

Implementation steps

1. **Understand the shared responsibility model:** Review the AWS shared responsibility model to understand your responsibilities for securing your workloads and data in the cloud. AWS is responsible for securing the underlying cloud infrastructure, while you are responsible for securing your applications, data, and the services you use.
2. **Implement vulnerability scanning**: Configure a vulnerability scanning service, such as Amazon Inspector, to automatically scan your compute instances (for example, virtual machines, containers, or serverless functions) for software vulnerabilities, potential defects, and unintended network exposure.
3. **Establish vulnerability management processes:** Define processes and procedures to identify, prioritize, and remediate vulnerabilities. This may include the setup of regular vulnerability scanning schedules, establishment of risk assessment criteria, and definition of remediation timelines based on vulnerability severity.
4. **Set up patch management:** Use a patch management service to automate the process of patching your compute instances, both for operating systems and applications. You can configure the service to scan instances for missing patches and automatically install them on a schedule. Consider AWS Systems Manager Patch Manager to provide this functionality.
5. **Configure malware protection:** Implement mechanisms to detect malicious software in your environment. For example, you can use tools like Amazon GuardDuty to analyze, detect, and alert of malware in EC2 and EBS volumes. GuardDuty can also scan newly uploaded objects to Amazon S3 for potential malware or viruses and take action to isolate them before they are ingested into downstream processes.
6. **Integrate vulnerability scanning in CI/CD pipelines:** If you're using a CI/CD pipeline for your application deployment, integrate vulnerability scanning tools into your pipeline. Tools like Amazon CodeGuru Security and open-source options can scan your source code, dependencies, and artifacts for potential security issues.
7. **Configure a security monitoring service:** Set up a security monitoring service, such as AWS Security Hub CSPM, to get a comprehensive view of your security posture across multiple cloud services. The service should collect security findings from various sources and present them in a standardized format for easier prioritization and remediation.
8. **Implement web application penetration testing**: If your application is a web application, and your organization has the necessary skills or can hire outside assistance, consider implementing web application penetration testing to identify potential vulnerabilities in your application.

9. **Automate with infrastructure as code**: Use infrastructure as code (IaC) tools, such as AWS CloudFormation, to automate the deployment and configuration of your resources, including the security services mentioned previously. This practice helps you create a more consistent and standardized resource architecture across multiple accounts and environments.
10. **Monitor and continually improve**: Continually monitor your vulnerability management program's effectiveness, and make improvements as needed. Review security findings, assess the effectiveness of your remediation efforts, and adjust your processes and tools accordingly.

Resources

Related documents:

- AWS Systems Manager
- Security Overview of AWS Lambda
- Amazon CodeGuru
- Improved, Automated Vulnerability Management for Cloud Workloads with a New Amazon Inspector
- Automate vulnerability management and remediation in AWS using Amazon Inspector and AWS Systems Manager – Part 1

Related videos:

- Securing Serverless and Container Services
- Security best practices for the Amazon EC2 instance metadata service

SEC06-BP02 Provision compute from hardened images

Provide fewer opportunities for unintended access to your runtime environments by deploying them from hardened images. Only acquire runtime dependencies, such as container images and application libraries, from trusted registries and verify their signatures. Create your own private registries to store trusted images and libraries for use in your build and deploy processes.

Desired outcome: Your compute resources are provisioned from hardened baseline images. You retrieve external dependencies, such as container images and application libraries, only from trusted registries and verify their signatures. These are stored in private registries for your build

and deployment processes to reference. You scan and update images and dependencies regularly to help protect against any newly discovered vulnerabilities.

Common anti-patterns:

- Acquiring images and libraries from trusted registries, but not verifying their signature or performing vulnerability scans before putting into use.
- Hardening images, but not regularly testing them for new vulnerabilities or updating to the latest version.
- Installing or not removing software packages that are not required during the expected lifecycle of the image.
- Relying solely on patching to keep production compute resources up to date. Patching alone can still cause compute resources to drift from the hardened standard over time. Patching can also fail to remove malware that may have been installed by a threat actor during a security event.

Benefits of establishing this best practice: Hardening images helps reduce the number of paths available in your runtime environment that can allow unintended access to unauthorized users or services. It also can reduce the scope of impact should any unintended access occur.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To harden your systems, start from the latest versions of operating systems, container images, and application libraries. Apply patches to known issues. Minimize the system by removing any unneeded applications, services, device drivers, default users, and other credentials. Take any other needed actions, such as disabling ports to create an environment that has only the resources and capabilities needed by your workloads. From this baseline, you can then install software, agents, or other processes you need for purposes such as workload monitoring or vulnerability management.

You can reduce the burden of hardening systems by using guidance that trusted sources provide, such as the Center for Internet Security (CIS) and the Defense Information Systems Agency (DISA) Security Technical Implementation Guides (STIGs). We recommend you start with an Amazon Machine Image (AMI) published by AWS or an APN partner, and use the AWS EC2 Image Builder to automate configuration according to an appropriate combination of CIS and STIG controls.

While there are available hardened images and EC2 Image Builder recipes that apply the CIS or DISA STIG recommendations, you may find their configuration prevents your software from running successfully. In this situation, you can start from a non-hardened base image, install your

software, and then incrementally apply CIS controls to test their impact. For any CIS control that prevents your software from running, test if you can implement the finer-grained hardening recommendations in a DISA instead. Keep track of the different CIS controls and DISA STIG configurations you are able to apply successfully. Use these to define your image hardening recipes in EC2 Image Builder accordingly.

For containerized workloads, hardened images from Docker are available on the Amazon Elastic Container Registry (ECR) public repository. You can use EC2 Image Builder to harden container images alongside AMIs.

Similar to operating systems and container images, you can obtain code packages (or *libraries*) from public repositories, through tooling such as pip, npm, Maven, and NuGet. We recommend you manage code packages by integrating private repositories, such as within AWS CodeArtifact, with trusted public repositories. This integration can handle retrieving, storing, and keeping packages up-to-date for you. Your application build processes can then obtain and test the latest version of these packages alongside your application, using techniques like Software Composition Analysis (SCA), Static Application Security Testing (SAST), and Dynamic Application Security Testing (DAST).

For serverless workloads that use AWS Lambda, simplify managing package dependencies using Lambda layers. Use Lambda layers to configure a set of standard dependencies that are shared across different functions into a standalone archive. You can create and maintain layers through their own build process, providing a central way for your functions to stay up-to-date.

Implementation steps

- Harden operating systems. Use base images from trusted sources as a foundation for building your hardened AMIs. Use EC2 Image Builder to help customize the software installed on your images.
- Harden containerized resources. Configure containerized resources to meet security best practices. When using containers, implement ECR Image Scanning in your build pipeline and on a regular basis against your image repository to look for CVEs in your containers.
- When using serverless implementation with AWS Lambda, use Lambda layers to segregate application function code and shared dependent libraries. Configure code signing for Lambda to make sure that only trusted code runs in your Lambda functions.

Resources

Related best practices:

- OPS05-BP05 Perform patch management

Related videos:

- Deep dive into AWS Lambda security

Related examples:

- Quickly build STIG-compliant AMI using EC2 Image Builder
- Building better container images
- Using Lambda layers to simplify your development process
- Develop & Deploy AWS Lambda Layers using Serverless Framework
- Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST and DAST tools

SEC06-BP03 Reduce manual management and interactive access

Use automation to perform deployment, configuration, maintenance, and investigative tasks wherever possible. Consider manual access to compute resources in cases of emergency procedures or in safe (sandbox) environments, when automation is not available.

Desired outcome: Programmatic scripts and automation documents (runbooks) capture authorized actions on your compute resources. These runbooks are initiated either automatically, through change detection systems, or manually, when human judgment is required. Direct access to compute resources is only made available in emergency situations when automation is not available. All manual activities are logged and incorporated into a review process to continually improve your automation capabilities.

Common anti-patterns:

- Interactive access to Amazon EC2 instances with protocols such as SSH or RDP.
- Maintaining individual user logins such as `/etc/passwd` or Windows local users.
- Sharing a password or private key to access an instance among multiple users.
- Manually installing software and creating or updating configuration files.
- Manually updating or patching software.
- Logging into an instance to troubleshoot problems.

Benefits of establishing this best practice: Performing actions with automation helps you to reduce the operational risk of unintended changes and misconfigurations. Removing the use of Secure Shell (SSH) and Remote Desktop Protocol (RDP) for interactive access reduces the scope of access to your compute resources. This takes away a common path for unauthorized actions. Capturing your compute resource management tasks in automation documents and programmatic scripts provides a mechanism to define and audit the full scope of authorized activities at a fine-grained level of detail.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Logging into an instance is a classic approach to system administration. After installing the server operating system, users would typically log in manually to configure the system and install the desired software. During the server's lifetime, users might log in to perform software updates, apply patches, change configurations, and troubleshoot problems.

Manual access poses a number of risks, however. It requires a server that listens for requests, such as an SSH or RDP service, that can provide a potential path to unauthorized access. It also increases the risk of human error associated with performing manual steps. These can result in workload incidents, data corruption or destruction, or other security issues. Human access also requires protections against the sharing of credentials, creating additional management overhead.

To mitigate these risks, you can implement an agent-based remote access solution, such as AWS Systems Manager. AWS Systems Manager Agent (SSM Agent) initiates an encrypted channel and thus does not rely on listening for externally-initiated requests. Consider configuring SSM Agent to establish this channel over a VPC endpoint.

Systems Manager gives you fine-grained control over how you can interact with your managed instances. You define the automations to run, who can run them, and when they can run. Systems Manager can apply patches, install software, and make configuration changes without interactive access to the instance. Systems Manager can also provide access to a remote shell and log every command invoked, and its output, during the session to logs and Amazon S3. AWS CloudTrail records invocations of Systems Manager APIs for inspection.

Implementation steps

1. Install AWS Systems Manager Agent (SSM Agent) on your Amazon EC2 instances. Check to see if SSM Agent is included and started automatically as part of your base AMI configuration.

2. Verify that the IAM Roles associated with your EC2 instance profiles include the `AmazonSSMManagedInstanceCore` managed IAM policy.
3. Disable SSH, RDP, and other remote access services running on your instances. You can do this by running scripts configured in the user data section of your launch templates or by building customized AMIs with tools such as EC2 Image Builder.
4. Verify that the security group ingress rules applicable to your EC2 instances do not permit access on port 22/tcp (SSH) or port 3389/tcp (RDP). Implement detection and alerting on misconfigured security groups using services such as AWS Config.
5. Define appropriate automations, runbooks, and run commands in Systems Manager. Use IAM policies to define who can perform these actions and the conditions under which they are permitted. Test these automations thoroughly in a non-production environment. Invoke these automations when necessary, instead of interactively accessing the instance.
6. Use AWS Systems Manager Session Manager to provide interactive access to instances when necessary. Turn on session activity logging to maintain an audit trail in Amazon CloudWatch Logs or Amazon S3.

Resources

Related best practices:

- REL08-BP04 Deploy using immutable infrastructure

Related examples:

- Replacing SSH access to reduce management and security overhead with AWS Systems Manager

Related tools:

- AWS Systems Manager

Related videos:

- Controlling User Session Access to Instances in AWS Systems Manager Session Manager

SEC06-BP04 Validate software integrity

Use cryptographic verification to validate the integrity of software artifacts (including images) your workload uses. Cryptographically sign your software as a safeguard against unauthorized changes run within your compute environments.

Desired outcome: All artifacts are obtained from trusted sources. Vendor website certificates are validated. Downloaded artifacts are cryptographically verified by their signatures. Your own software is cryptographically signed and verified by your computing environments.

Common anti-patterns:

- Trusting reputable vendor websites to obtain software artifacts, but ignoring certificate expiration notices. Proceeding with downloads without confirming certificates are valid.
- Validating vendor website certificates, but not cryptographically verifying downloaded artifacts from these websites.
- Relying solely on digests or hashes to validate software integrity. Hashes establish that artifacts have not been modified from the original version, but do not validate their source.
- Not signing your own software, code, or libraries, even when only used in your own deployments.

Benefits of establishing this best practice: Validating the integrity of artifacts that your workload depends on helps prevent malware from entering your compute environments. Signing your software helps safeguard against unauthorized running in your compute environments. Secure your software supply chain by signing and verifying code.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Operating system images, container images, and code artifacts are often distributed with integrity checks available, such as through a digest or hash. These allow clients to verify integrity by computing their own hash of the payload and validating it is the same as the one published. While these checks help verify that the payload has not been tampered with, they do not validate the payload came from the original source (its *provenance*). Verifying provenance requires a certificate that a trusted authority issues to digitally sign the artifact.

If you are using a downloaded software or artifacts in your workload, check if the provider provides a public key for digital signature verification. Here are some examples of how AWS provides a public key and verification instructions for software we publish:

- EC2 Image Builder: Verify the signature of the AWSTOE installation download
- AWS Systems Manager: Verifying the signature of SSM Agent
- Amazon CloudWatch: Verifying the signature of the CloudWatch agent package

Incorporate digital signature verification into the processes you use for obtaining and hardening images, as discussed in SEC06-BP02 Provision compute from hardened images.

You can use AWS Signer to help manage the verification of signatures, as well as your own code-signing lifecycle for your own software and artifacts. Both AWS Lambda and Amazon Elastic Container Registry provide integrations with Signer to verify the signatures of your code and images. Using the examples in the Resources section, you can incorporate Signer into your continuous integration and delivery (CI/CD) pipelines to automate verification of signatures and the signing of your own code and images.

Resources

Related documents:

- Cryptographic Signing for Containers
- Best Practices to help secure your container image build pipeline by using AWS Signer
- Announcing Container Image Signing with AWS Signer and Amazon EKS
- Configuring code signing for AWS Lambda
- Best practices and advanced patterns for Lambda code signing
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys

Related examples:

- Automate Lambda code signing with Amazon CodeCatalyst and AWS Signer
- Signing and Validating OCI Artifacts with AWS Signer

Related tools:

- AWS Lambda
- AWS Signer
- AWS Certificate Manager
- AWS Key Management Service
- AWS CodeArtifact

SEC06-BP05 Automate compute protection

Automate compute protection operations to reduce the need for human intervention. Use automated scanning to detect potential issues within your compute resources, and remediate with automated programmatic responses or fleet management operations. Incorporate automation in your CI/CD processes to deploy trustworthy workloads with up-to-date dependencies.

Desired outcome: Automated systems perform all scanning and patching of compute resources. You use automated verification to check that software images and dependencies come from trusted sources, and have not been tampered with. Workloads are automatically checked for up-to-date dependencies, and are signed to establish trustworthiness in AWS compute environments. Automated remediations are initiated when non-compliant resources are detected.

Common anti-patterns:

- Following the practice of immutable infrastructure, but not having a solution in place for emergency patching or replacement of production systems.
- Using automation to fix misconfigured resources, but not having a manual override mechanism in place. Situations may arise where you need to adjust the requirements, and you may need to suspend automations until you make these changes.

Benefits of establishing this best practice: Automation can reduce the risk of unauthorized access and use of your compute resources. It helps to prevent misconfigurations from making their way into production environments, and detecting and fixing misconfigurations should they occur. Automation also helps to detect unauthorized access and use of compute resources to reduce your time to respond. This in turn can reduce the overall scope of impact from the issue.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

You can apply the automations described in the Security Pillar practices for protecting your compute resources. SEC06-BP01 Perform vulnerability management describes how you can use Amazon Inspector in both your CI/CD pipelines and for continually scanning your runtime environments for known Common Vulnerabilities and Exposures (CVEs). You can use AWS Systems Manager to apply patches or redeploy from fresh images through automated runbooks to keep your compute fleet updated with the latest software and libraries. Use these techniques to reduce the need for manual processes and interactive access to your compute resources. See SEC06-BP03 Reduce manual management and interactive access to learn more.

Automation also plays a role in deploying workloads that are trustworthy, described in SEC06-BP02 Provision compute from hardened images and SEC06-BP04 Validate software integrity. You can use services such as EC2 Image Builder, AWS Signer, AWS CodeArtifact, and Amazon Elastic Container Registry (ECR) to download, verify, construct, and store hardened and approved images and code dependencies. Alongside Inspector, each of these can play a role in your CI/CD process so your workload makes its way to production only when it is confirmed that its dependencies are up-to-date and from trusted sources. Your workload is also signed so AWS compute environments, such as AWS Lambda and Amazon Elastic Kubernetes Service (EKS) can verify it hasn't been tampered with before allowing it to run.

Beyond these preventative controls, you can use automation in your detective controls for your compute resources as well. As one example, AWS Security Hub CSPM offers the NIST 800-53 Rev. 5 standard that includes checks such as [EC2.8] EC2 instances should use Instance Metadata Service Version 2 (IMDSv2). IMDSv2 uses the techniques of session authentication, blocking requests that contain an X-Forwarded-For HTTP header, and a network TTL of 1 to stop traffic originating from external sources to retrieve information about the EC2 instance. This check in Security Hub CSPM can detect when EC2 instances use IMDSv1 and initiate automated remediation. Learn more about automated detection and remediations in SEC04-BP04 Initiate remediation for non-compliant resources.

Implementation steps

1. Automate creating secure, compliant and hardened AMIs with EC2 Image Builder. You can produce images that incorporate controls from the Center for Internet Security (CIS) Benchmarks or Security Technical Implementation Guide (STIG) standards from base AWS and APN partner images.

2. Automate configuration management. Enforce and validate secure configurations in your compute resources automatically by using a configuration management service or tool.
   a. Automated configuration management using AWS Config
   b. Automated security and compliance posture management using AWS Security Hub CSPM
3. Automate patching or replacing Amazon Elastic Compute Cloud (Amazon EC2) instances. AWS Systems Manager Patch Manager automates the process of patching managed instances with both security-related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications.
   a. AWS Systems Manager Patch Manager
4. Automate scanning of compute resources for common vulnerabilities and exposures (CVEs), and embed security scanning solutions within your build pipeline.
   a. Amazon Inspector
   b. ECR Image Scanning
5. Consider Amazon GuardDuty for automatic malware and threat detection to protect compute resources. GuardDuty can also identify potential issues when an AWS Lambda function gets invoked in your AWS environment.
   a. Amazon GuardDuty
6. Consider AWS Partner solutions. AWS Partners offer industry-leading products that are equivalent, identical to, or integrate with existing controls in your on-premises environments. These products complement the existing AWS services to allow you to deploy a comprehensive security architecture and a more seamless experience across your cloud and on-premises environments.
   a. Infrastructure security

Resources

Related best practices:

- SEC01-BP06 Automate deployment of standard security controls

Related documents:

- Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure

Related videos:

- Security best practices for the Amazon EC2 instance metadata service

Data protection

Questions

- SEC 7. How do you classify your data?
- SEC 8. How do you protect your data at rest?
- SEC 9. How do you protect your data in transit?

SEC 7. How do you classify your data?

Classification provides a way to categorize data, based on criticality and sensitivity in order to help you determine appropriate protection and retention controls.

Best practices

- SEC07-BP01 Understand your data classification scheme
- SEC07-BP02 Apply data protection controls based on data sensitivity
- SEC07-BP03 Automate identification and classification
- SEC07-BP04 Define scalable data lifecycle management

SEC07-BP01 Understand your data classification scheme

Understand the classification of data your workload is processing, its handling requirements, the associated business processes, where the data is stored, and who the data owner is. Your data classification and handling scheme should consider the applicable legal and compliance requirements of your workload and what data controls are needed. Understanding the data is the first step in the data classification journey.

Desired outcome: The types of data present in your workload are well-understood and documented. Appropriate controls are in place to protect sensitive data based on its classification. These controls govern considerations such as who is allowed to access the data and for what purpose, where the data is stored, the encryption policy for that data and how encryption keys are managed, the lifecycle for the data and its retention requirements, appropriate destruction processes, what backup and recovery processes are in place, and the auditing of access.

Common anti-patterns:

- Not having a formal data classification policy in place to define data sensitivity levels and their handling requirements
- Not having a good understanding of the sensitivity levels of data within your workload, and not capturing this information in architecture and operations documentation
- Failing to apply the appropriate controls around your data based on its sensitivity and requirements, as outlined in your data classification and handling policy
- Failing to provide feedback about data classification and handling requirements to owners of the policies.

Benefits of establishing this best practice: This practice removes ambiguity around the appropriate handling of data within your workload. Applying a formal policy that defines the sensitivity levels of data in your organization and their required protections can help you comply with legal regulations and other cybersecurity attestations and certifications. Workload owners can have confidence in knowing where sensitive data is stored and what protection controls are in place. Capturing these in documentation helps new team members better understand them and maintain controls early in their tenure. These practices can also help reduce costs by right sizing the controls for each type of data.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When designing a workload, you may be considering ways to protect sensitive data intuitively. For example, in a multi-tenant application, it is intuitive to think of each tenant's data as sensitive and put protections in place so that one tenant can't access the data of another tenant. Likewise, you may intuitively design access controls so only administrators can modify data while other users have only read-level access or no access at all.

By having these data sensitivity levels defined and captured in policy, along with their data protection requirements, you can formally identify what data resides in your workload. You can then determine if the right controls are in place, if the controls can be audited, and what responses are appropriate if data is found to be mishandled.

To help identify where sensitive data resides within your workload, consider using a data catalog. A data catalog is a database that maps data in your organization, its location, sensitivity level,

and the controls in place to protect that data. Additionally, consider using resource tags where available. For example, you can apply a tag that has a *tag key* of `Classification` and a *tag value* of `PHI` for protected health information (PHI), and another tag that has a *tag key* of `Sensitivity` and a *tag value* of `High`. Services such as AWS Config can then be used to monitor these resources for changes and alert if they are modified in a way that brings them out of compliance with your protection requirements (such as changing the encryption settings). You can capture the standard definition of your tag keys and acceptable values using tag policies, a feature of AWS Organizations. It is not recommended that the tag key or value contains private or sensitive data.

Implementation steps

1. Understand your organization's data classification scheme and protection requirements.
2. Identify the types of sensitive data processed by your workloads.
3. Capture the data in a data catalog that provides a single view of where data resides in the organization and the level of sensitivity of that data.
4. Consider using resource and data-level tagging, where available, to tag data with its sensitivity level and other operational metadata that can help with monitoring and incident response.
    a. AWS Organizations tag policies can be used to enforce tagging standards.

Resources

Related best practices:

- SUS04-BP01 Implement a data classification policy

Related documents:

- Data Classification whitepaper
- Best Practices for Tagging AWS Resources

Related examples:

- AWS Organizations Tag Policy Syntax and Examples

Related tools

- AWS Tag Editor

SEC07-BP02 Apply data protection controls based on data sensitivity

Apply data protection controls that provide an appropriate level of control for each class of data defined in your classification policy. This practice can allow you to protect sensitive data from unauthorized access and use, while preserving the availability and use of data.

Desired outcome: You have a classification policy that defines the different levels of sensitivity for data in your organization. For each of these sensitivity levels, you have clear guidelines published for approved storage and handling services and locations, and their required configuration. You implement the controls for each level according to the level of protection required and their associated costs. You have monitoring and alerting in place to detect if data is present in unauthorized locations, processed in unauthorized environments, accessed by unauthorized actors, or the configuration of related services becomes non-compliant.

Common anti-patterns:

- Applying the same level of protection controls across all data. This may lead to over-provisioning security controls for low-sensitivity data, or insufficient protection of highly sensitive data.
- Not involving relevant stakeholders from security, compliance, and business teams when defining data protection controls.
- Overlooking the operational overhead and costs associated with implementing and maintaining data protection controls.
- Not conducting periodic data protection control reviews to maintain alignment with classification policies.
- Not having a complete inventory of where data resides at rest and in transit.

Benefits of establishing this best practice: By aligning your controls to the classification level of your data, your organization can invest in higher levels of control where needed. This can include increasing resources on securing, monitoring, measuring, remediating, and reporting. Where fewer controls are appropriate, you can improve the accessibility and completeness of data for your workforce, customers, or constituents. This approach gives your organization the most flexibility with data usage, while still adhering to data protection requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing data protection controls based on data sensitivity levels involves several key steps. First, identify the different data sensitivity levels within your workload architecture (such as public, internal, confidential, and restricted) and evaluate where you store and process this data. Next, define isolation boundaries around data based on its sensitivity level. We recommend you separate data into different AWS accounts, using service control policies (SCPs) to restrict services and actions allowed for each data sensitivity level. This way, you can create strong isolation boundaries and enforce the principle of least privilege.

After you define the isolation boundaries, implement appropriate protection controls based on the data sensitivity levels. Refer to best practices for Protecting data at rest and Protecting data in transit to implement relevant controls like encryption, access controls, and auditing. Consider techniques like tokenization or anonymization to reduce the sensitivity level of your data. Simplify applying consistent data policies across your business with a centralized system for tokenization and de-tokenization.

Continuously monitor and test the effectiveness of the implemented controls. Regularly review and update the data classification scheme, risk assessments, and protection controls as your organization's data landscape and threats evolve. Align the implemented data protection controls with relevant industry regulations, standards, and legal requirements. Further, provide security awareness and training to help employees understand the data classification scheme and their responsibilities in handling and protecting sensitive data.

Implementation steps

1. Identify the classification and sensitivity levels of data within your workload.
2. Define isolation boundaries for each level and determine an enforcement strategy.
3. Evaluate the controls you define that govern access, encryption, auditing, retention, and others required by your data classification policy.
4. Evaluate options to reduce the sensitivity level of data where appropriate, such as using tokenization or anonymization.
5. Verify your controls using automated testing and monitoring of your configured resources.

Resources

Related best practices:

- PERF03-BP01 Use a purpose-built data store that best supports your data access and storage requirements
- COST04-BP05 Enforce data retention policies

Related documents:

- Data Classification whitepaper
- Best Practices for Security, Identify, & Compliance
- AWS KMS Best Practices
- Encryption best practices and features for AWS services

Related examples:

- Building a serverless tokenization solution to mask sensitive data
- How to use tokenization to improve data security and reduce audit scope

Related tools:

- AWS Key Management Service (AWS KMS)
- AWS CloudHSM
- AWS Organizations

SEC07-BP03 Automate identification and classification

Automating the identification and classification of data can help you implement the correct controls. Using automation to augment manual determination reduces the risk of human error and exposure.

Desired outcome: You are able to verify whether the proper controls are in place based on your classification and handling policy. Automated tools and services help you to identify and classify the sensitivity level of your data. Automation also helps you continually monitor your environments to detect and alert if data is being stored or handled in unauthorized ways so corrective action can be taken quickly.

Common anti-patterns:

- Relying solely on manual processes for data identification and classification, which can be error-prone and time-consuming. This can lead to inefficient and inconsistent data classification, especially as data volumes grow.
- Not having mechanisms to track and manage data assets across the organization.
- Overlooking the need for continuous monitoring and classification of data as it moves and evolves within the organization.

Benefits of establishing this best practice: Automating data identification and classification can lead to more consistent and accurate application of data protection controls, reducing the risk of human error. Automation can also provide visibility into sensitive data access and movement, helping you detect unauthorized handling and take corrective action.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

While human judgment is often used to classify data during the initial design phases of a workload, consider having systems in place that automate identification and classification on test data as a preventive control. For example, developers can be provided a tool or service to scan representative data to determine its sensitivity. Within AWS, you can upload data sets into Amazon S3 and scan them using Amazon Macie, Amazon Comprehend, or Amazon Comprehend Medical. Likewise, consider scanning data as part of unit and integration testing to detect where sensitive data is not expected. Alerting on sensitive data at this stage can highlight gaps in protections before deployment to production. Other features such as sensitive data detection in AWS Glue, Amazon SNS, and Amazon CloudWatch can also be used to detect PII and take mitigating action. For any automated tool or service, understand how it defines sensitive data, and augment it with other human or automated solutions to close any gaps as needed.

As a detective control, use ongoing monitoring of your environments to detect if sensitive data is being stored in non-compliant ways. This can help detect situations such as sensitive data being emitted into log files or being copied to a data analytics environment without proper de-identification or redaction. Data that is stored in Amazon S3 can be continually monitored for sensitive data using Amazon Macie.

Implementation steps

1. Review the data classification scheme within your organization described in SEC07-BP01.

   a. With an understanding of your organization's data classification scheme, you can establish accurate processes for automated identification and classification that align with your company's policies.

2. Perform an initial scan of your environments for automated identification and classification.
   a. An initial full scan of your data can help produce a comprehensive understanding of where sensitive data resides in your environments. When a full scan is not initially required or is unable to be completed up-front due to cost, evaluate if data sampling techniques are suitable to achieve your outcomes. For example, Amazon Macie can be configured to perform a broad automated sensitive data discovery operation across your S3 buckets. This capability uses sampling techniques to cost-efficiently perform a preliminary analysis of where sensitive data resides. A deeper analysis of S3 buckets can then be performed using a sensitive data discovery job. Other data stores can also be exported to S3 to be scanned by Macie.
   b. Establish access control defined in SEC07-BP02 for your data storage resources identified within your scan.
3. Configure ongoing scans of your environments.
   a. The automated sensitive data discovery capability of Macie can be used to perform ongoing scans of your environments. Known S3 buckets that are authorized to store sensitive data can be excluded using an allow list in Macie.
4. Incorporate identification and classification into your build and test processes.
   a. Identify tools that developers can use to scan data for sensitivity while workloads are in development. Use these tools as part of integration testing to alert when sensitive data is unexpected and prevent further deployment.
5. Implement a system or runbook to take action when sensitive data is found in unauthorized locations.
   a. Restrict access to data using auto-remediation. For example, you can move this data to an S3 bucket with restricted access or tag the object if you use attribute-based access control (ABAC). Additionally, consider masking the data when it is detected.
   b. Alert your data protection and incident response teams to investigate the root cause of the incident. Any learnings they identify can help prevent future incidents.

Resources

Related documents:

- AWS Glue: Detect and process sensitive data

- Using managed data identifiers in Amazon SNS
- Amazon CloudWatch Logs: Help protect sensitive log data with masking

Related examples:

- Enabling data classification for Amazon RDS database with Macie
- Detecting sensitive data in DynamoDB with Macie

Related tools:

- Amazon Macie
- Amazon Comprehend
- Amazon Comprehend Medical
- AWS Glue

SEC07-BP04 Define scalable data lifecycle management

Understand your data lifecycle requirements as they relate to your different levels of data classification and handling. This can include how data is handled when it first enters your environment, how data is transformed, and the rules for its destruction. Consider factors such as retention periods, access, auditing, and tracking provenance.

Desired outcome: You classify data as close as possible to the point and time of ingestion. When data classification requires masking, tokenization, or other processes that reduce sensitivity level, you perform these actions as close as possible to point and time of ingestion.

You delete data in accordance with your policy when it is no longer appropriate to keep, based on its classification.

Common anti-patterns:

- Implementing a one-size-fits-all approach to data lifecycle management, without considering varying sensitivity levels and access requirements.
- Considering lifecycle management only from the perspective of either data that is usable, or data that is backed up, but not both.
- Assuming that data that has entered your workload is valid, without establishing its value or provenance.

- Relying on data durability as a substitute for data backups and protection.
- Retaining data beyond its usefulness and required retention period.

Benefits of establishing this best practice: A well-defined and scalable data lifecycle management strategy helps maintain regulatory compliance, improves data security, optimizes storage costs, and enables efficient data access and sharing while maintaining appropriate controls.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Data within a workload is often dynamic. The form it takes when entering your workload environment can be different from when it is stored or used in business logic, reporting, analytics, or machine learning. In addition, the value of data can change over time. Some data is temporal in nature and loses value as it gets older. Consider how these changes to your data impact evaluation under your data classification scheme and associated controls. Where possible, use an automated lifecycle mechanism, such as Amazon S3 lifecycle policies and the Amazon Data Lifecycle Manager, to configure your data retention, archiving, and expiration processes. For data stored in DynamoDB, you can use the Time To Live (TTL) feature to define a per-item expiration timestamp.

Distinguish between data that is available for use, and data that is stored as a backup. Consider using AWS Backup to automate the backup of data across AWS services. Amazon EBS snapshots provide a way to copy an EBS volume and store it using S3 features, including lifecycle, data protection, and access to protection mechanisms. Two of these mechanisms are S3 Object Lock and AWS Backup Vault Lock, which can provide you with additional security and control over your backups. Manage clear separation of duties and access for backups. Isolate backups at the account level to maintain separation from the affected environment during an event.

Another aspect of lifecycle management is recording the history of data as it progresses through your workload, called *data provenance tracking*. This can give confidence that you know where the data came from, any transformations performed, what owner or process made those changes, and when. Having this history helps with troubleshooting issues and investigations during potential security events. For example, you can log metadata about transformations in an Amazon DynamoDB table. Within a data lake, you can keep copies of transformed data in different S3 buckets for each data pipeline stage. Store schema and timestamp information in an AWS Glue Data Catalog. Regardless of your solution, consider the requirements of your end users to determine the appropriate tooling you need to report on your data provenance. This will help you determine how to best track your provenance.

Implementation steps

1. Analyze the workload's data types, sensitivity levels, and access requirements to classify the data and define appropriate lifecycle management strategies.
2. Design and implement data retention policies and automated destruction processes that align with legal, regulatory, and organizational requirements.
3. Establish processes and automation for continuous monitoring, auditing, and adjustment of data lifecycle management strategies, controls, and policies as workload requirements and regulations evolve.
    a. Detect resources that do not have automated lifecycle management turned on with AWS Config

Resources

Related best practices:

- COST04-BP05 Enforce data retention policies
- SUS04-BP03 Use policies to manage the lifecycle of your datasets

Related documents:

- Data Classification Whitepaper
- AWS Blueprint for Ransomware Defense
- DevOps Guidance: Improve traceability with data provenance tracking

Related examples:

- How to protect sensitive data for its entire lifecycle in AWS
- Build data lineage for data lakes using AWS Glue, Amazon Neptune, and Spline

Related tools:

- AWS Backup
- Amazon Data Lifecycle Manager
- AWS Identity and Access Management Access Analyzer

SEC 8. How do you protect your data at rest?

Protect your data at rest by implementing multiple controls, to reduce the risk of unauthorized access or mishandling.

Best practices

- SEC08-BP01 Implement secure key management
- SEC08-BP02 Enforce encryption at rest
- SEC08-BP03 Automate data at rest protection
- SEC08-BP04 Enforce access control

SEC08-BP01 Implement secure key management

Secure key management includes the storage, rotation, access control, and monitoring of key material required to secure data at rest for your workload.

Desired outcome: You have a scalable, repeatable, and automated key management mechanism. The mechanism enforces least privilege access to key material and provides the correct balance between key availability, confidentiality, and integrity. You monitor access to the keys, and if rotation of key material is required, you rotate them using an automated process. You do not allow key material to be accessed by human operators.

Common anti-patterns:

- Human access to unencrypted key material.
- Creating custom cryptographic algorithms.
- Overly broad permissions to access key material.

Benefits of establishing this best practice: By establishing a secure key management mechanism for your workload, you can help provide protection for your content against unauthorized access. Additionally, you may be subject to regulatory requirements to encrypt your data. An effective key management solution can provide technical mechanisms aligned to those regulations to protect key material.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Encryption of data at rest is a fundamental security control. To implement this control, your workload needs a mechanism to securely store and manage the key material used to encrypt your data at rest.

AWS offers the AWS Key Management Service (AWS KMS) to provide durable, secure, and redundant storage for AWS KMS keys. Many AWS services integrate with AWS KMS to support encryption of your data. AWS KMS uses FIPS 140-3 Level 3 validated hardware security modules to protect your keys. There is no mechanism to export AWS KMS keys in plain text.

When deploying workloads using a multi-account strategy, you should keep AWS KMS keys in the same account as the workload that uses them. This distributed model places the responsibility for managing the AWS KMS keys with your team. In other use cases, your organization may choose to store AWS KMS keys into a centralized account. This centralized structure requires additional policies to enable the cross-account access required for the workload account to access keys stored in the centralized account, but may be more applicable in use cases where a single key is shared across multiple AWS accounts.

Regardless of where the key material is stored, you should tightly control access to the key through the use of key policies and IAM policies. Key policies are the primary way to control access to an AWS KMS key. Additionally, AWS KMS key grants can provide access to AWS services to encrypt and decrypt data on your behalf. Review the guidance for access control to your AWS KMS keys.

You should monitor the use of encryption keys to detect unusual access patterns. Operations performed using AWS managed keys and customer managed keys stored in AWS KMS can be logged in AWS CloudTrail and should be reviewed periodically. Pay special attention to monitoring key destruction events. To mitigate accidental or malicious destruction of key material, key destruction events do not delete the key material immediately. Attempts to delete keys in AWS KMS are subject to a waiting period, which defaults to 30 days and a minimum of 7 days, providing administrators time to review these actions and roll back the request if necessary.

Most AWS services use AWS KMS in a way that is transparent to you - your only requirement is to decide whether to use an AWS managed or customer managed key. If your workload requires the direct use of AWS KMS to encrypt or decrypt data, you should use envelope encryption to protect your data. The AWS Encryption SDK can provide your applications client-side encryption primitives to implement envelope encryption and integrate with AWS KMS.

Implementation steps

1. Determine the appropriate key management options (AWS managed or customer managed) for the key.
   a. For ease of use, AWS offers AWS owned and AWS managed keys for most services, which provide encryption-at-rest capability without the need to manage key material or key policies.
   b. When using customer managed keys, consider the default key store to provide the best balance between agility, security, data sovereignty, and availability. Other use cases may require the use of custom key stores with AWS CloudHSM or the external key store.
2. Review the list of services that you are using for your workload to understand how AWS KMS integrates with the service. For example, EC2 instances can use encrypted EBS volumes, verifying that Amazon EBS snapshots created from those volumes are also encrypted using a customer managed key and mitigating accidental disclosure of unencrypted snapshot data.
   a. How AWS services use AWS KMS
   b. For detailed information about the encryption options that an AWS service offers, see the Encryption at Rest topic in the user guide or the developer guide for the service.
3. Implement AWS KMS: AWS KMS makes it simple for you to create and manage keys and control the use of encryption across a wide range of AWS services and in your applications.
   a. Getting started: AWS Key Management Service (AWS KMS)
   b. Review the best practices for access control to your AWS KMS keys.
4. Consider AWS Encryption SDK: Use the AWS Encryption SDK with AWS KMS integration when your application needs to encrypt data client-side.
   a. AWS Encryption SDK
5. Enable IAM Access Analyzer to automatically review and notify if there are overly broad AWS KMS key policies.
   a. Consider using custom policy checks to verify that a resource policy update does not grant public access to KMS Keys.
6. Enable Security Hub CSPM to receive notifications if there are misconfigured key policies, keys scheduled for deletion, or keys without automated rotation enabled.
7. Determine the logging level appropriate for your AWS KMS keys. Since calls to AWS KMS, including read-only events, are logged, the CloudTrail logs associated with AWS KMS can become voluminous.

a. Some organizations prefer to segregate the AWS KMS logging activity into a separate trail. For more detail, see the Logging AWS KMS API calls with CloudTrail section of the AWS KMS developers guide.

Resources

Related documents:

- AWS Key Management Service
- AWS cryptographic services and tools
- Protecting Amazon S3 Data Using Encryption
- Envelope encryption
- Digital sovereignty pledge
- Demystifying AWS KMS key operations, bring your own key, custom key store, and ciphertext portability
- AWS Key Management Service cryptographic details

Related videos:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS
- AWS data protection: Using locks, keys, signatures, and certificates

Related examples:

- Implement advanced access control mechanisms using AWS KMS

SEC08-BP02 Enforce encryption at rest

Encrypt private data at rest to maintain confidentiality and provide an additional layer of protection against unintended data disclosure or exfiltration. Encryption protects data so that it cannot be read or accessed without first being decrypted. Inventory and control unencrypted data to mitigate risks associated with data exposure.

Desired outcome: You have mechanisms that encrypt private data by default when at rest. These mechanisms help maintain the confidentiality of the data and provides an additional layer

of protection against unintended data disclosure or exfiltration. You maintain an inventory of unencrypted data and understand the controls that are in place to protect it.

Common anti-patterns:

- Not using encrypt-by-default configurations.
- Providing overly permissive access to decryption keys.
- Not monitoring the use of encryption and decryption keys.
- Storing data unencrypted.
- Using the same encryption key for all data regardless of data usage, types, and classification.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Map encryption keys to data classifications within your workloads. This approach helps protect against overly permissive access when using either a single, or very small number of encryption keys for your data (see SEC07-BP01 Understand your data classification scheme).

AWS Key Management Service (AWS KMS) integrates with many AWS services to make it easier to encrypt your data at rest. For example, in Amazon Elastic Compute Cloud (Amazon EC2), you can set default encryption on accounts so that new EBS volumes are automatically encrypted. When using AWS KMS, consider how tightly the data needs to be restricted. Default and service-controlled AWS KMS keys are managed and used on your behalf by AWS. For sensitive data that requires fine-grained access to the underlying encryption key, consider customer managed keys (CMKs). You have full control over CMKs, including rotation and access management through the use of key policies.

Additionally, services such as Amazon Simple Storage Service (Amazon S3) now encrypt all new objects by default. This implementation provides enhanced security with no impact on performance.

Other services, such as Amazon Elastic Compute Cloud (Amazon EC2) or Amazon Elastic File System (Amazon EFS), support settings for default encryption. You can also use AWS Config Rules to automatically check that you are using encryption for Amazon Elastic Block Store (Amazon EBS) volumes, Amazon Relational Database Service (Amazon RDS) instances, Amazon S3 buckets, and other services within your organization.

AWS also provides options for client-side encryption, allowing you to encrypt data prior to uploading it to the cloud. The AWS Encryption SDK provides a way to encrypt your data using envelope encryption. You provide the wrapping key, and the AWS Encryption SDK generates a unique data key for each data object it encrypts. Consider AWS CloudHSM if you need a managed single-tenant hardware security module (HSM). AWS CloudHSM allows you to generate, import, and manage cryptographic keys on a FIPS 140-2 level 3 validated HSM. Some use cases for AWS CloudHSM include protecting private keys for issuing a certificate authority (CA), and turning on transparent data encryption (TDE) for Oracle databases. The AWS CloudHSM Client SDK provides software that allows you to encrypt data client side using keys stored inside AWS CloudHSM prior to uploading your data into AWS. The Amazon DynamoDB Encryption Client also allows you to encrypt and sign items prior to upload into a DynamoDB table.

Implementation steps

- **Configure default encryption for new Amazon EBS volumes:** Specify that you want all newly created Amazon EBS volumes to be created in encrypted form, with the option of using the default key provided by AWS or a key that you create.
- **Configure encrypted Amazon Machine Images (AMIs):** Copying an existing AMI with encryption configured will automatically encrypt root volumes and snapshots.
- **Configure Amazon RDS encryption:** Configure encryption for your Amazon RDS database clusters and snapshots at rest by using the encryption option.
- **Create and configure AWS KMS keys with policies that limit access to the appropriate principals for each classification of data:** For example, create one AWS KMS key for encrypting production data and a different key for encrypting development or test data. You can also provide key access to other AWS accounts. Consider having different accounts for your development and production environments. If your production environment needs to decrypt artifacts in the development account, you can edit the CMK policy used to encrypt the development artifacts to give the production account the ability to decrypt those artifacts. The production environment can then ingest the decrypted data for use in production.
- **Configure encryption in additional AWS services:** For other AWS services you use, review the security documentation for that service to determine the service's encryption options.

Resources

Related documents:

- AWS Crypto Tools

- AWS Encryption SDK
- AWS KMS Cryptographic Details Whitepaper
- AWS Key Management Service
- AWS cryptographic services and tools
- Amazon EBS Encryption
- Default encryption for Amazon EBS volumes
- Encrypting Amazon RDS Resources
- How do I enable default encryption for an Amazon S3 bucket?
- Protecting Amazon S3 Data Using Encryption

Related videos:

- How Encryption Works in AWS
- Securing Your Block Storage on AWS

SEC08-BP03 Automate data at rest protection

Use automation to validate and enforce data at rest controls. Use automated scanning to detect misconfiguration of your data storage solutions, and perform remediations through automated programmatic response where possible. Incorporate automation in your CI/CD processes to detect data storage misconfigurations before they are deployed to production.

Desired outcome: Automated systems scan and monitor data storage locations for misconfiguration of controls, unauthorized access, and unexpected use. Detection of misconfigured storage locations initiates automated remediations. Automated processes create data backups and store immutable copies outside of the original environment.

Common anti-patterns:

- Not considering options to enable encryption by default settings, where supported.
- Not considering security events, in addition to operational events, when formulating an automated backup and recovery strategy.
- Not enforcing public access settings for storage services.
- Not monitoring and audit your controls for protecting data at rest.

Benefits of establishing this best practice: Automation helps to prevent the risk of misconfiguring your data storage locations. It helps to prevent misconfigurations from entering your production environments. This best practice also helps with detecting and fixing misconfigurations if they occur.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Automation is a theme throughout the practices for protecting your data at rest. SEC01-BP06 Automate deployment of standard security controls describes how you can capture the configuration of your resources using *infrastructure as code* (IaC) templates, such as with AWS CloudFormation. These templates are committed to a version control system, and are used to deploy resources on AWS through a CI/CD pipeline. These techniques equally apply to automating the configuration of your data storage solutions, such as encryption settings on Amazon S3 buckets.

You can check the settings that you define in your IaC templates for misconfiguration in your CI/CD pipelines using rules in AWS CloudFormation Guard. You can monitor settings that are not yet available in CloudFormation or other IaC tooling for misconfiguration with AWS Config. Alerts that Config generates for misconfigurations can be remediated automatically, as described in SEC04-BP04 Initiate remediation for non-compliant resources.

Using automation as part of your permissions management strategy is also an integral component of automated data protections. SEC03-BP02 Grant least privilege access and SEC03-BP04 Reduce permissions continuously describe configuring least-privilege access policies that are continually monitored by the AWS Identity and Access Management Access Analyzer to generate findings when permission can be reduced. Beyond automation for monitoring permissions, you can configure Amazon GuardDuty to watch for anomalous data access behavior for your EBS volumes (by way of an EC2 instance), S3 buckets, and supported Amazon Relational Database Service databases.

Automation also plays a role in detecting when sensitive data is stored in unauthorized locations. SEC07-BP03 Automate identification and classification describes how Amazon Macie can monitor your S3 buckets for unexpected sensitive data and generate alerts that can initiate an automated response.

Follow the practices in REL09 Back up data to develop an automated data backup and recovery strategy. Data backup and recovery is as important for recovering from security events as it is for operational events.

Implementation steps

1. Capture data storage configuration in IaC templates. Use automated checks in your CI/CD pipelines to detect misconfigurations.
   a. You can use for CloudFormation your IaC templates, and CloudFormation Guard for checking templates for misconfiguration.
   b. Use AWS Config to run rules in a proactive evaluation mode. Use this setting to check the compliance of a resource as a step in your CI/CD pipeline before creating it.
2. Monitor resources for data storage misconfigurations.
   a. Set AWS Config to monitor data storage resources for changes in control configurations and generate alerts to invoke remediation actions when it detects a misconfiguration.
   b. See SEC04-BP04 Initiate remediation for non-compliant resources for more guidance on automated remediations.
3. Monitor and reduce data access permissions continually through automation.
   a. IAM Access Analyzer can run continually to generate alerts when permissions can potentially be reduced.
4. Monitor and alert on anomalous data access behaviors.
   a. GuardDuty watches for both known threat signatures and deviations from baseline access behaviors for data storage resources such as EBS volumes, S3 buckets, and RDS databases.
5. Monitor and alert on sensitive data being stored in unexpected locations.
   a. Use Amazon Macie to continually scan your S3 buckets for sensitive data.
6. Automate secure and encrypted backups of your data.
   a. AWS Backup is a managed service that creates encrypted and secure backups of various data sources on AWS. Elastic Disaster Recovery allows you to copy full server workloads and maintain continuous data protection with a recovery point objective (RPO) measured in seconds. You can configure both services to work together to automate creating data backups and copying them to failover locations. This can help keep your data available when impacted by either operational or security events.

Resources

Related best practices:

- SEC01-BP06 Automate deployment of standard security controls

- SEC03-BP02 Grant least privilege access
- SEC03-BP04 Reduce permissions continuously
- SEC04-BP04 Initiate remediation for non-compliant resources
- SEC07-BP03 Automate identification and classification
- REL09-BP02 Secure and encrypt backups
- REL09-BP03 Perform data backup automatically

Related documents:

- AWS Prescriptive Guidance: Automatically encrypt existing and new Amazon EBS volumes
- Ransomware Risk Management on AWS Using the NIST Cyber Security Framework (CSF)

Related examples:

- How to use AWS Config proactive rules and AWS CloudFormation Hooks to prevent creation of noncompliant cloud resources
- Automate and centrally manage data protection for Amazon S3 with AWS Backup
- AWS re:Invent 2023 - Implement proactive data protection using Amazon EBS snapshots
- AWS re:Invent 2022 - Build and automate for resilience with modern data protection

Related tools:

- AWS CloudFormation Guard
- AWS CloudFormation Guard Rules Registry
- IAM Access Analyzer
- Amazon Macie
- AWS Backup
- Elastic Disaster Recovery

SEC08-BP04 Enforce access control

To help protect your data at rest, enforce access control using mechanisms such as isolation and versioning. Apply least privilege and conditional access controls. Prevent granting public access to your data.

Desired outcome: You verify that only authorized users can access data on a need-to-know basis. You protect your data with regular backups and versioning to prevent against intentional or inadvertent modification or deletion of data. You isolate critical data from other data to protect its confidentiality and data integrity.

Common anti-patterns:

- Storing data with different sensitivity requirements or classification together.
- Using overly permissive permissions on decryption keys.
- Improperly classifying data.
- Not retaining detailed backups of important data.
- Providing persistent access to production data.
- Not auditing data access or regularly reviewing permissions.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Protecting data at rest is important to maintain data integrity, confidentiality, and compliance with regulatory requirements. You can implement multiple controls to help achieve this, including access control, isolation, conditional access, and versioning.

You can enforce access control with the principle of least privilege, which provides only the necessary permissions to users and services to perform their tasks. This includes access to encryption keys. Review your AWS Key Management Service (AWS KMS) policies to verify that the level of access you grant is appropriate and that relevant conditions apply.

You can separate data based on different classification levels by using distinct AWS accounts for each level, and manage these accounts using AWS Organizations. This isolation can help prevent unauthorized access and minimizes the risk of data exposure.

Regularly review the level of access granted in Amazon S3 bucket policies. Avoid using publicly readable or writeable buckets unless absolutely necessary. Consider using AWS Config to detect publicly available buckets and Amazon CloudFront to serve content from Amazon S3. Verify that buckets that should not allow public access are properly configured to prevent it.

Implement versioning and object locking mechanisms for critical data stored in Amazon S3. Amazon S3 versioning preserves previous versions of objects to recover data from accidental

deletion or overwrites. Amazon S3 Object Lock provides mandatory access control for objects, which prevents them from being deleted or overwritten, even by the root user, until the lock expires. Additionally, Amazon Glacier Vault Lock offers a similar feature for archives stored in Amazon Glacier.

Implementation steps

1. **Enforce access control with the principle of least privilege**:
   - Review the access permissions granted to users and services, and verify that they have only the necessary permissions to perform their tasks.
   - Review access to encryption keys by checking the AWS Key Management Service (AWS KMS) policies.
2. **Separate data based on different classification levels**:
   - Use distinct AWS accounts for each data classification level.
   - Manage these accounts using AWS Organizations.
3. **Review Amazon S3 bucket and object permissions**:
   - Regularly review the level of access granted in Amazon S3 bucket policies.
   - Avoid using publicly readable or writeable buckets unless absolutely necessary.
   - Consider using AWS Config to detect publicly available buckets.
   - Use Amazon CloudFront to serve content from Amazon S3.
   - Verify that buckets that should not allow public access are properly configured to prevent it.
   - You can apply the same review process for databases and any other data sources that use IAM authentication, such as SQS or third-party data stores.
4. **Use AWS IAM Access Analyzer**:
   - You can configure AWS IAM Access Analyzer to analyze Amazon S3 buckets and generate findings when an S3 policy grants access to an external entity.
5. **Implement versioning and object locking mechanisms**:
   - Use Amazon S3 versioning to preserve previous versions of objects, which provides recovery from accidental deletion or overwrites.
   - Use Amazon S3 Object Lock to provide mandatory access control for objects, which prevents them from being deleted or overwritten, even by the root user, until the lock expires.
   - Use Amazon Glacier Vault Lock for archives stored in Amazon Glacier.
6. **Use Amazon S3 Inventory**:

- You can use Amazon S3 Inventory to audit and report on the replication and encryption status of your S3 objects.

7. **Review Amazon EBS and AMI sharing permissions**:
   - Review your sharing permissions for Amazon EBS and AMI sharing to verify that your images and volumes are not shared with AWS accounts that are external to your workload.
8. **Review AWS Resource Access Manager Shares periodically**:
   - You can use AWS Resource Access Manager to share resources, such as AWS Network Firewall policies, Amazon Route 53 resolver rules, and subnets, within your Amazon VPCs.
   - Audit shared resources regularly and stop sharing resources that no longer need to be shared.

Resources

Related best practices:

- SEC03-BP01 Define access requirements
- SEC03-BP02 Grant least privilege access

Related documents:

- AWS KMS Cryptographic Details Whitepaper
- Introduction to Managing Access Permissions to Your Amazon S3 Resources
- Overview of managing access to your AWS KMS resources
- AWS Config Rules
- Amazon S3 + Amazon CloudFront: A Match Made in the Cloud
- Using versioning
- Locking Objects Using Amazon S3 Object Lock
- Sharing an Amazon EBS Snapshot
- Shared AMIs
- Hosting a single-page application on Amazon S3
- AWS Global Condition Keys
- Building a Data Perimeter on AWS

Related videos:

- Securing Your Block Storage on AWS

SEC 9. How do you protect your data in transit?

Protect your data in transit by implementing multiple controls to reduce the risk of unauthorized access or loss.

Best practices

- SEC09-BP01 Implement secure key and certificate management
- SEC09-BP02 Enforce encryption in transit
- SEC09-BP03 Authenticate network communications

SEC09-BP01 Implement secure key and certificate management

Transport Layer Security (TLS) certificates are used to secure network communications and establish the identity of websites, resources, and workloads over the internet, as well as private networks.

Desired outcome: A secure certificate management system that can provision, deploy, store, and renew certificates in a public key infrastructure (PKI). A secure key and certificate management mechanism prevents certificate private key material from disclosure and automatically renews the certificate on a periodic basis. It also integrates with other services to provide secure network communications and identity for machine resources inside of your workload. Key material should never be accessible to human identities.

Common anti-patterns:

- Performing manual steps during the certificate deployment or renewal processes.
- Paying insufficient attention to certificate authority (CA) hierarchy when designing a private CA.
- Using self-signed certificates for public resources.

Benefits of establishing this best practice:

- Simplify certificate management through automated deployment and renewal
- Encourage encryption of data in transit using TLS certificates
- Increased security and auditability of certificate actions taken by the certificate authority
- Organization of management duties at different layers of the CA hierarchy

Level of risk exposed if this best practice is not established: High

Implementation guidance

Modern workloads make extensive use of encrypted network communications using PKI protocols such as TLS. PKI certificate management can be complex, but automated certificate provisioning, deployment, and renewal can reduce the friction associated with certificate management.

AWS provides two services to manage general-purpose PKI certificates: AWS Certificate Manager and AWS Private Certificate Authority (AWS Private CA). ACM is the primary service that customers use to provision, manage, and deploy certificates for use in both public-facing as well as private AWS workloads. ACM issues private certificates using AWS Private CA and integrates with many other AWS managed services to provide secure TLS certificates for workloads. ACM can also issue publicly trusted certificates from Amazon Trust Services. Public certificates from ACM can be used on public facing workloads, as modern browsers and operating systems trust these certificates by default.

AWS Private CA allows you to establish your own root or subordinate certificate authority and issue TLS certificates through an API. You can use these kinds of certificates in scenarios where you control and manage the trust chain on the client side of the TLS connection. In addition to TLS use cases, AWS Private CA can be used to issue certificates to Kubernetes pods, Matter device product attestations, code signing, and other use cases with a custom template. You can also use IAM Roles Anywhere to provide temporary IAM credentials to on-premises workloads that have been issued X.509 certificates signed by your Private CA.

In addition to ACM and AWS Private CA, AWS IoT Core provides specialized support for provisioning, managing and deploying PKI certificates to IoT devices. AWS IoT Core provides specialized mechanisms for onboarding IoT devices into your public key infrastructure at scale.

Some AWS services, such as Amazon API Gateway and Elastic Load Balancing, offer their own capabilities for using certificates to secure application connections. For example, both API Gateway and Application Load Balancer (ALB) support mutual TLS (mTLS) using client certificates that you create and export using the AWS Management Console, CLI, or APIs.

Considerations for establishing a private CA hierarchy

When you need to establish a private CA, it's important to take special care to properly design the CA hierarchy upfront. It's a best practice to deploy each level of your CA hierarchy into separate AWS accounts when creating a private CA hierarchy. This intentional step reduces the surface area

for each level in the CA hierarchy, making it simpler to discover anomalies in CloudTrail log data and reducing the scope of access or impact if there is unauthorized access to one of the accounts. The root CA should reside in its own separate account and should only be used to issue one or more intermediate CA certificates.

Then, create one or more intermediate CAs in accounts separate from the root CA's account to issue certificates for end users, devices, or other workloads. Finally, issue certificates from your root CA to the intermediate CAs, which will in turn issue certificates to your end users or devices. For more information on planning your CA deployment and designing your CA hierarchy, including planning for resiliency, cross-region replication, sharing CAs across your organization, and more, see Planning your AWS Private CA deployment.

Implementation steps

1. Determine the relevant AWS services required for your use case:
   - Many use cases can leverage the existing AWS public key infrastructure using AWS Certificate Manager. ACM can be used to deploy TLS certificates for web servers, load balancers, or other uses for publicly trusted certificates.
   - Consider AWS Private CA when you need to establish your own private certificate authority hierarchy or need access to exportable certificates. ACM can then be used to issue many types of end-entity certificates using the AWS Private CA.
   - For use cases where certificates must be provisioned at scale for embedded Internet of things (IoT) devices, consider AWS IoT Core.
   - Consider using native mTLS functionality in services like Amazon API Gateway or Application Load Balancer.
2. Implement automated certificate renewal whenever possible:
   - Use ACM managed renewal for certificates issued by ACM along with integrated AWS managed services.
3. Establish logging and audit trails:
   - Enable CloudTrail logs to track access to the accounts holding certificate authorities. Consider configuring log file integrity validation in CloudTrail to verify the authenticity of the log data.
   - Periodically generate and review audit reports that list the certificates that your private CA has issued or revoked. These reports can be exported to an S3 bucket.
   - When deploying a private CA, you will also need to establish an S3 bucket to store the Certificate Revocation List (CRL). For guidance on configuring this S3 bucket based on your workload's requirements, see Planning a certificate revocation list (CRL).

Resources

Related best practices:

- SEC02-BP02 Use temporary credentials
- SEC08-BP01 Implement secure key management
- SEC09-BP03 Authenticate network communications

Related documents:

- How to host and manage an entire private certificate infrastructure in AWS
- How to secure an enterprise scale ACM Private CA hierarchy for automotive and manufacturing
- Private CA best practices
- How to use AWS RAM to share your ACM Private CA cross-account

Related videos:

- Activating AWS Certificate Manager Private CA (workshop)

Related examples:

- Private CA workshop
- IOT Device Management Workshop (including device provisioning)

Related tools:

- Plugin to Kubernetes cert-manager to use AWS Private CA

SEC09-BP02 Enforce encryption in transit

Enforce your defined encryption requirements based on your organization's policies, regulatory obligations and standards to help meet organizational, legal, and compliance requirements. Only use protocols with encryption when transmitting sensitive data outside of your virtual private cloud (VPC). Encryption helps maintain data confidentiality even when the data transits untrusted networks.

Desired outcome: You encrypt network traffic between your resources and the internet to mitigate unauthorized access to the data. You encrypt network traffic within your internal AWS environment according to your security requirements. You encrypt data in transit using secure TLS protocols and cipher suites.

Common anti-patterns:

- Using deprecated versions of SSL, TLS, and cipher suite components (for example, SSL v3.0, 1024-bit RSA keys, and RC4 cipher).
- Allowing unencrypted (HTTP) traffic to or from public-facing resources.
- Not monitoring and replacing X.509 certificates prior to expiration.
- Using self-signed X.509 certificates for TLS.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS services provide HTTPS endpoints using TLS for communication, providing encryption in transit when communicating with the AWS APIs. Insecure HTTP protocols can be audited and blocked in a Virtual Private Cloud (VPC) through the use of security groups. HTTP requests can also be automatically redirected to HTTPS in Amazon CloudFront or on an Application Load Balancer. You can use an Amazon Simple Storage Service (Amazon S3) bucket policy to restrict the ability to upload objects through HTTP, effectively enforcing the use of HTTPS for object uploads to your bucket(s). You have full control over your computing resources to implement encryption in transit across your services. Additionally, you can use VPN connectivity into your VPC from an external network or AWS Direct Connect to facilitate encryption of traffic. Verify that your clients make calls to AWS APIs using at least TLS 1.2, as AWS has deprecated the use of earlier versions of TLS as of February 2024. We recommend you use TLS 1.3. If you have special requirements for encryption in transit, you can find third-party solutions available in the AWS Marketplace.

Implementation steps

- **Enforce encryption in transit:** Your defined encryption requirements should be based on the latest standards and best practices and only allow secure protocols. For example, configure a security group to only allow the HTTPS protocol to an application load balancer or Amazon EC2 instance.
- **Configure secure protocols in edge services:** Configure HTTPS with Amazon CloudFront and use a security profile appropriate for your security posture and use case.

- **Use a VPN for external connectivity:** Consider using an IPsec VPN for securing point-to-point or network-to-network connections to help provide both data privacy and integrity.
- **Configure secure protocols in load balancers:** Select a security policy that provides the strongest cipher suites supported by the clients that will be connecting to the listener. Create an HTTPS listener for your Application Load Balancer.
- **Configure secure protocols in Amazon Redshift:** Configure your cluster to require a secure socket layer (SSL) or transport layer security (TLS) connection.
- **Configure secure protocols:** Review AWS service documentation to determine encryption-in-transit capabilities.
- **Configure secure access when uploading to Amazon S3 buckets:** Use Amazon S3 bucket policy controls to enforce secure access to data.
- **Consider using AWS Certificate Manager:** ACM allows you to provision, manage, and deploy public TLS certificates for use with AWS services.
- **Consider using AWS Private Certificate Authority for private PKI needs:** AWS Private CA allows you to create private certificate authority (CA) hierarchies to issue end-entity X.509 certificates that can be used to create encrypted TLS channels.

Resources

Related documents:

- Using HTTPS with CloudFront
- Connect your VPC to remote networks using AWS Virtual Private Network
- Create an HTTPS listener for your Application Load Balancer
- Tutorial: Configure SSL/TLS on Amazon Linux 2
- Using SSL/TLS to encrypt a connection to a DB instance
- Configuring security options for connections

SEC09-BP03 Authenticate network communications

Verify the identity of communications by using protocols that support authentication, such as Transport Layer Security (TLS) or IPsec.

Design your workload to use secure, authenticated network protocols whenever communicating between services, applications, or to users. Using network protocols that support authentication

and authorization provides stronger control over network flows and reduces the impact of unauthorized access.

Desired outcome: A workload with well-defined data plane and control plane traffic flows between services. The traffic flows use authenticated and encrypted network protocols where technically feasible.

Common anti-patterns:

- Unencrypted or unauthenticated traffic flows within your workload.
- Reusing authentication credentials across multiple users or entities.
- Relying solely on network controls as an access control mechanism.
- Creating a custom authentication mechanism rather than relying on industry-standard authentication mechanisms.
- Overly permissive traffic flows between service components or other resources in the VPC.

Benefits of establishing this best practice:

- Limits the scope of impact for unauthorized access to one part of the workload.
- Provides a higher level of assurance that actions are only performed by authenticated entities.
- Improves decoupling of services by clearly defining and enforcing intended data transfer interfaces.
- Enhances monitoring, logging, and incident response through request attribution and well-defined communication interfaces.
- Provides defense-in-depth for your workloads by combining network controls with authentication and authorization controls.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Your workload's network traffic patterns can be characterized into two categories:

- *East-west traffic* represents traffic flows between services that make up a workload.
- *North-south traffic* represents traffic flows between your workload and consumers.

While it is common practice to encrypt north-south traffic, securing east-west traffic using authenticated protocols is less common. Modern security practices recommend that network design alone does not grant a trusted relationship between two entities. When two services may reside within a common network boundary, it is still best practice to encrypt, authenticate, and authorize communications between those services.

As an example, AWS service APIs use the AWS Signature Version 4 (SigV4) signature protocol to authenticate the caller, no matter what network the request originates from. This authentication ensures that AWS APIs can verify the identity that requested the action, and that identity can then be combined with policies to make an authorization decision to determine whether the action should be allowed or not.

Services such as Amazon VPC Lattice and Amazon API Gateway allow you use the same SigV4 signature protocol to add authentication and authorization to east-west traffic in your own workloads. If resources outside of your AWS environment need to communicate with services that require SigV4-based authentication and authorization, you can use AWS Identity and Access Management (IAM) Roles Anywhere on the non-AWS resource to acquire temporary AWS credentials. These credentials can be used to sign requests to services using SigV4 to authorize access.

Another common mechanism for authenticating east-west traffic is TLS mutual authentication (mTLS). Many Internet of Things (IoT), business-to-business applications, and microservices use mTLS to validate the identity of both sides of a TLS communication through the use of both client and server-side X.509 certificates. These certificates can be issued by AWS Private Certificate Authority (AWS Private CA). You can use services such as Amazon API Gateway to provide mTLS authentication for inter- or intra-workload communication. Application Load Balancer also supports mTLS for internal or external facing workloads. While mTLS provides authentication information for both sides of a TLS communication, it does not provide a mechanism for authorization.

Finally, OAuth 2.0 and OpenID Connect (OIDC) are two protocols typically used for controlling access to services by users, but are now becoming popular for service-to-service traffic as well. API Gateway provides a JSON Web Token (JWT) authorizer, allowing workloads to restrict access to API routes using JWTs issued from OIDC or OAuth 2.0 identity providers. OAuth2 scopes can be used as a source for basic authorization decisions, but the authorization checks still need to be implemented in the application layer, and OAuth2 scopes alone cannot support more complex authorization needs.

Implementation steps

- **Define and document your workload network flows:** The first step in implementing a defense-in-depth strategy is defining your workload's traffic flows.
  - Create a data flow diagram that clearly defines how data is transmitted between different services that comprise your workload. This diagram is the first step to enforcing those flows through authenticated network channels.
  - Instrument your workload in development and testing phases to validate that the data flow diagram accurately reflects the workload's behavior at runtime.
  - A data flow diagram can also be useful when performing a threat modeling exercise, as described in SEC01-BP07 Identify threats and prioritize mitigations using a threat model.
- **Establish network controls:** Consider AWS capabilities to establish network controls aligned to your data flows. While network boundaries should not be the only security control, they provide a layer in the defense-in-depth strategy to protect your workload.
  - Use security groups to establish define and restrict data flows between resources.
  - Consider using AWS PrivateLink to communicate with both AWS and third-party services that support AWS PrivateLink. Data sent through a AWS PrivateLink interface endpoint stays within the AWS network backbone and does not traverse the public Internet.
- **Implement authentication and authorization across services in your workload:** Choose the set of AWS services most appropriate to provide authenticated, encrypted traffic flows in your workload.
  - Consider Amazon VPC Lattice to secure service-to-service communication. VPC Lattice can use SigV4 authentication combined with auth policies to control service-to-service access.
  - For service-to-service communication using mTLS, consider API Gateway, Application Load Balancer. AWS Private CA can be used to establish a private CA hierarchy capable of issuing certificates for use with mTLS.
  - When integrating with services using OAuth 2.0 or OIDC, consider API Gateway using the JWT authorizer.
  - For communication between your workload and IoT devices, consider AWS IoT Core, which provides several options for network traffic encryption and authentication.
- **Monitor for unauthorized access:** Continually monitor for unintended communication channels, unauthorized principals attempting to access protected resources, and other improper access patterns.

- If using VPC Lattice to manage access to your services, consider enabling and monitoring VPC Lattice access logs. These access logs include information on the requesting entity, network information including source and destination VPC, and request metadata.
- Consider enabling VPC flow logs to capture metadata on network flows and periodically review for anomalies.
- Refer to the AWS Security Incident Response Guide and the Incident Response section of the AWS Well-Architected Framework security pillar for more guidance on planning, simulating, and responding to security incidents.

Resources

Related best practices:

- SEC03-BP07 Analyze public and cross-account access
- SEC02-BP02 Use temporary credentials
- SEC01-BP07 Identify threats and prioritize mitigations using a threat model

Related documents:

- Evaluating access control methods to secure Amazon API Gateway APIs
- Configuring mutual TLS authentication for a REST API
- How to secure API Gateway HTTP endpoints with JWT authorizer
- Authorizing direct calls to AWS services using AWS IoT Core credential provider
- AWS Security Incident Response Guide

Related videos:

- AWS re:invent 2022: Introducing VPC Lattice
- AWS re:invent 2020: Serverless API authentication for HTTP APIs on AWS

Related examples:

- Amazon VPC Lattice Workshop
- Zero-Trust Episode 1 – The Phantom Service Perimeter workshop

Incident response

Question

- SEC 10. How do you anticipate, respond to, and recover from incidents?

SEC 10. How do you anticipate, respond to, and recover from incidents?

Even with mature preventive and detective controls, your organization should implement mechanisms to respond to and mitigate the potential impact of security incidents. Your preparation strongly affects the ability of your teams to operate effectively during an incident, to isolate, contain and perform forensics on issues, and to restore operations to a known good state. Putting in place the tools and access ahead of a security incident, then routinely practicing incident response through game days, helps ensure that you can recover while minimizing business disruption.

Best practices

- SEC10-BP01 Identify key personnel and external resources
- SEC10-BP02 Develop incident management plans
- SEC10-BP03 Prepare forensic capabilities
- SEC10-BP04 Develop and test security incident response playbooks
- SEC10-BP05 Pre-provision access
- SEC10-BP06 Pre-deploy tools
- SEC10-BP07 Run simulations
- SEC10-BP08 Establish a framework for learning from incidents

SEC10-BP01 Identify key personnel and external resources

Identify internal and external personnel, resources, and legal obligations to help your organization respond to an incident.

Desired outcome: You have a list of key personnel, their contact information, and the roles they play when responding to a security event. You review this information regularly and update it to reflect personnel changes from an internal and external tools perspective. You consider all third-party service providers and vendors while documenting this information, including security partners, cloud providers, and software-as-a-service (SaaS) applications. During a security event,

personnel are available with the appropriate level of responsibility, context, and access to be able to respond and recover.

Common anti-patterns:

- Not maintaining an updated list of key personnel with contact information, their roles, and their responsibilities when responding to security events.
- Assuming that everyone understands the people, dependencies, infrastructure, and solutions when responding to and recovering from an event.
- Not having a document or knowledge repository that represents key infrastructure or application design.
- Not having proper onboarding processes for new employees to effectively contribute to a security event response, such as conducting event simulations.
- Not having an escalation path in place when key personnel are temporarily unavailable or fail to respond during security events.

Benefits of establishing this best practice: This practice reduces the triage and response time spent on identifying the right personnel and their roles during an event. Minimize wasted time during an event by maintaining an updated list of key personnel and their roles so you can bring the right individuals to triage and recover from an event.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify key personnel in your organization: Maintain a contact list of personnel within your organization that you need to involve. Regularly review and update this information in the event of personnel movement, like organizational changes, promotions, and team changes. This is especially important for key roles like incident managers, incident responders, and communications lead.

- **Incident manager:** Incident managers have overall authority during the event response.
- **Incident responders:** Incident responders are responsible for investigation and remediation activities. These people can differ based on the type of event, but are typically developers and operation teams responsible for the impacted application.
- **Communications lead:** The communications lead is responsible for internal and external communications, especially with public agencies, regulators, and customers.

- **Onboarding process:** Regularly train and onboard new employees to equip them with the necessary skills and knowledge to contribute effectively to incident response efforts. Incorporate simulations and hands-on exercises as part of the onboarding process to facilitate their preparedness
- **Subject matter experts (SMEs):** In the case of distributed and autonomous teams, we recommend you identify an SME for mission critical workloads. They offer insights into the operation and data classification of critical workloads involved in the event.

Example table format:

```
  | Role | Name | Contact Information | Responsibilities |
1 | ——- | ——- | ——- | ——- |
2 | Incident Manager | Jane Doe| jane.doe@example.com | Overall authority during
 response |
3 | Incident Responder | John Smith | john.smith@example.com | Investigation and
 remediation |
4 | Communications Lead | Emily Johnson | emily.johnson@example.com | Internal and
 external communications |
5 | Communications Lead | Michael Brown | michael.brown@example.com | Insights on
 critical workloads |
```

Consider using the AWS Systems Manager Incident Manager feature to capture key contacts, define a response plan, automate on-call schedules, and create escalation plans. Automate and rotate all staff through an on-call schedule, so that responsibility for the workload is shared across its owners. This promotes good practices, such as emitting relevant metrics and logs as well as defining alarm thresholds that matter for the workload.

Identify external partners: Enterprises use tools built by independent software vendors (ISVs), partners, and subcontractors to build differentiating solutions for their customers. Engage key personnel from these parties who can help respond to and recover from an incident. We recommend you sign up for the appropriate level of Support in order to get prompt access to AWS subject matter experts through a support case. Consider similar arrangements with all critical solutions providers for the workloads. Some security events require publicly listed businesses to notify relevant public agencies and regulators of the event and impacts. Maintain and update contact information for the relevant departments and responsible individuals.

Implementation steps

1. Set up an incident management solution.

   a. Consider deploying Incident Manager in your Security Tooling account.

2. Define contacts in your incident management solution.
   a. Define at least two types of contact channels for each contact (such as SMS, phone, or email), to ensure reachability during an incident.
3. Define a response plan.
   a. Identify the most appropriate contacts to engage during an incident. Define escalation plans aligned to the roles of personnel to be engaged, rather than individual contacts. Consider including contacts that may be responsible for informing external entities, even if they are not directly engaged to resolve the incident.

Resources

Related best practices:

- OPS02-BP03 Operations activities have identified owners responsible for their performance

Related documents:

- AWS Security Incident Response Guide

Related examples:

- AWS customer playbook framework
- Prepare for and respond to security incidents in your AWS environment

Related tools:

- AWS Systems Manager Incident Manager

Related videos:

- Amazon's approach to security during development

SEC10-BP02 Develop incident management plans

The first document to develop for incident response is the incident response plan. The incident response plan is designed to be the foundation for your incident response program and strategy.

Benefits of establishing this best practice: Developing thorough and clearly defined incident response processes is key to a successful and scalable incident response program. When a security event occurs, clear steps and workflows can help you to respond in a timely manner. You might already have existing incident response processes. Regardless of your current state, it's important to update, iterate, and test your incident response processes regularly.

Level of risk exposed if this best practice is not established: High

Implementation guidance

An incident management plan is critical to respond, mitigate, and recover from the potential impact of security incidents. An incident management plan is a structured process for identifying, remediating, and responding in a timely matter to security incidents.

The cloud has many of the same operational roles and requirements found in an on-premises environment. When you create an incident management plan, it is important to factor response and recovery strategies that best align with your business outcome and compliance requirements. For example, if you operate workloads in AWS that are FedRAMP compliant in the United States, follow the recommendations in NIST SP 800-61 Computer Security Handling Guide. Similarly, when you operate workloads that store personally identifiable information (PII), consider how to protect and respond to issues related to data residency and use.

When building an incident management plan for your workloads in AWS, start with the AWS Shared Responsibility Model for building a defense-in-depth approach towards incident response. In this model, AWS manages security of the cloud, and you are responsible for security in the cloud. This means that you retain control and are responsible for the security controls you choose to implement. The AWS Security Incident Response Guide details key concepts and foundational guidance for building a cloud-centric incident management plan.

An effective incident management plan must be continually iterated upon, remaining current with your cloud operations goal. Consider using the implementation plans detailed below as you create and evolve your incident management plan.

Implementation steps

1. Define roles and responsibilities within your organization for handling security events. This should involve representatives from various departments, including:
   - Human resources (HR)
   - Executive team
   - Legal department
   - Application owners and developers (subject matter experts, or SMEs)
2. Clearly outline who is responsible, accountable, consulted, and informed (RACI) during an incident. Create a RACI chart to facilitate quick and direct communication, and clearly outline the leadership across different stages of an event.
3. Involve application owners and developers (SMEs) during an incident, as they can provide valuable information and context to aid in measuring the impact. Build relationships with these SMEs, and practice incident response scenarios with them before an actual incident occurs.
4. Involve trusted partners or external experts in the investigation or response process, as they can provide additional expertise and perspective.
5. Align your incident management plans and roles with any local regulations or compliance requirements that govern your organization.
6. Practice and test your incident response plans regularly, and involve all the defined roles and responsibilities. This helps streamline the process and verify you have a coordinated and efficient response to security incidents.
7. Review and update the roles, responsibilities, and RACI chart periodically, or as your organizational structure or requirements change.

Understand AWS response teams and support

- **AWS Support**
  - Support offers a range of plans that provide access to tools and expertise that support the success and operational health of your AWS solutions. If you need technical support and more resources to help plan, deploy, and optimize your AWS environment, you can select a support plan that best aligns with your AWS use case.
  - Consider the Support Center in AWS Management Console (sign-in required) as the central point of contact to get support for issues that affect your AWS resources. Access to Support

is controlled by AWS Identity and Access Management. For more information about getting access to Support features, see Getting started with Support.

- **AWS Customer Incident Response Team (CIRT)**
  - The AWS Customer Incident Response Team (CIRT) is a specialized 24/7 global AWS team that provides support to customers during active security events on the customer side of the AWS Shared Responsibility Model.
  - When the AWS CIRT supports you, they provide assistance with triage and recovery for an active security event on AWS. They can assist in root cause analysis through the use of AWS service logs and provide you with recommendations for recovery. They can also provide security recommendations and best practices to help you avoid security events in the future.
  - AWS customers can engage the AWS CIRT through an Support case.
- **AWS Security Incident Response**
  - Announced at re:Invent 2024, AWS Security Incident Response is a managed security incident response service that uses both modern triage technology and a human in the loop. The service ingests all GuardDuty findings and any third-party findings sent to AWS Security Hub CSPM for triage to alert the customer only on findings that require an investigation. The service also provides a portal to submit reactive cases in the event of a security event the customer notices and receive support from AWS' advanced incident response team.
- **DDoS response support**
  - AWS offers AWS Shield, which provides a managed distributed denial of service (DDoS) protection service that safeguards web applications running on AWS. Shield provides always-on detection and automatic inline mitigations that can minimize application downtime and latency, so there is no need to engage Support to benefit from DDoS protection. There are two tiers of Shield: AWS Shield Standard and AWS Shield Advanced. To learn about the differences between these two tiers, see Shield features documentation.
- **AWS Managed Services (AMS)**
  - AWS Managed Services (AMS) provides ongoing management of your AWS infrastructure so you can focus on your applications. By implementing best practices to maintain your infrastructure, AMS helps reduce your operational overhead and risk. AMS automates common activities such as change requests, monitoring, patch management, security, and backup services, and provides full-lifecycle services to provision, run, and support your infrastructure.
  - AMS takes responsibility for deploying a suite of security detective controls and provides a 24/7 first line of response to alerts. When an alert is initiated, AMS follows a standard set of automated and manual playbooks to verify a consistent response. These playbooks are shared

with AMS customers during onboarding so that they can develop and coordinate a response with AMS.

Develop the incident response plan

The incident response plan is designed to be the foundation for your incident response program and strategy. The incident response plan should be in a formal document. An incident response plan typically includes these sections:

- **An incident response team overview:** Outlines the goals and functions of the incident response team.
- **Roles and responsibilities:** Lists the incident response stakeholders and details their roles when an incident occurs.
- **A communication plan:** Details contact information and how you communicate during an incident.
- **Backup communication methods:** It's a best practice to have out-of-band communication as a backup for incident communication. An example of an application that provides a secure out-of-band communications channel is AWS Wickr.
- **Phases of incident response and actions to take:** Enumerates the phases of incident response (for example, detect, analyze, eradicate, contain, and recover), including high-level actions to take within those phases.
- **Incident severity and prioritization definitions:** Details how to classify the severity of an incident, how to prioritize the incident, and then how the severity definitions affect escalation procedures.

While these sections are common throughout companies of different sizes and industries, each organization's incident response plan is unique. You need to build an incident response plan that works best for your organization.

Resources

Related best practices:

- SEC04 Detection

Related documents:

- AWS Security Incident Response Guide
- NIST: Computer Security Incident Handling Guide

SEC10-BP03 Prepare forensic capabilities

Ahead of a security incident, consider developing forensics capabilities to support security event investigations.

Level of risk exposed if this best practice is not established: Medium

Concepts from traditional on-premises forensics apply to AWS. For key information to start building forensics capabilities in the AWS Cloud, see Forensic investigation environment strategies in the AWS Cloud.

Once you have your environment and AWS account structure set up for forensics, define the technologies required to effectively perform forensically sound methodologies across the four phases:

- **Collection:** Collect relevant AWS logs, such as AWS CloudTrail, AWS Config, VPC Flow Logs, and host-level logs. Collect snapshots, backups, and memory dumps of impacted AWS resources where available.
- **Examination:** Examine the data collected by extracting and assessing the relevant information.
- **Analysis:** Analyze the data collected in order to understand the incident and draw conclusions from it.
- **Reporting:** Present the information resulting from the analysis phase.

Implementation steps

Prepare your forensics environment

AWS Organizations helps you centrally manage and govern an AWS environment as you grow and scale AWS resources. An AWS organization consolidates your AWS accounts so that you can administer them as a single unit. You can use organizational units (OUs) to group accounts together to administer as a single unit.

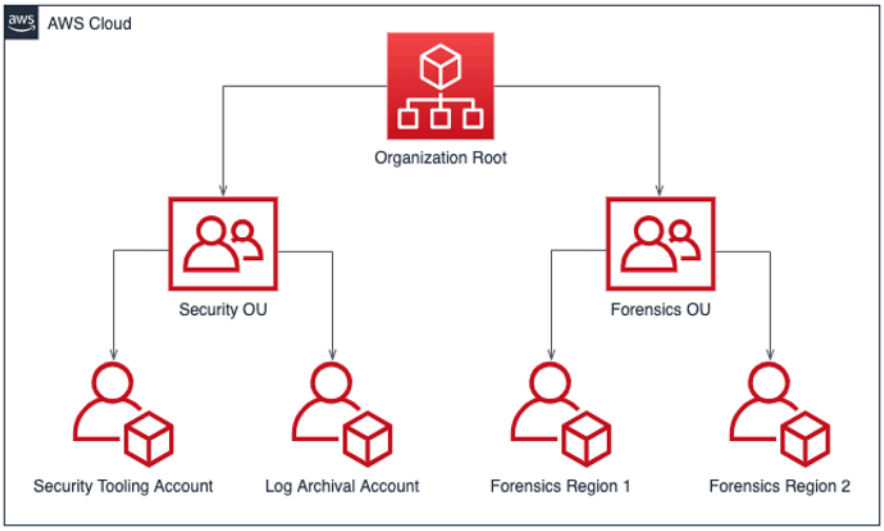
For incident response, it's helpful to have an AWS account structure that supports the functions of incident response, which includes a *security OU* and a *forensics OU*. Within the security OU, you should have accounts for:

- **Log archival:** Aggregate logs in a log archival AWS account with limited permissions.
- **Security tools:** Centralize security services in a security tool AWS account. This account operates as the delegated administrator for security services.

Within the forensics OU, you have the option to implement a single forensics account or accounts for each Region that you operate in, depending on which works best for your business and operational model. If you create a forensics account per Region, you can block the creation of AWS resources outside of that Region and reduce the risk of resources being copied to an unintended region. For example, if you only operate in US East (N. Virginia) Region (`us-east-1`) and US West (Oregon) (`us-west-2`), then you would have two accounts in the forensics OU: one for `us-east-1` and one for `us-west-2`.

You can create a forensics AWS account for multiple Regions. You should exercise caution in copying AWS resources to that account to verify you're aligning with your data sovereignty requirements. Because it takes time to provision new accounts, it is imperative to create and instrument the forensics accounts well ahead of an incident so that responders can be prepared to effectively use them for response.

The following diagram displays a sample account structure including a forensics OU with per-Region forensics accounts:



Per-Region account structure for incident response

Capture backups and snapshots

Setting up backups of key systems and databases are critical for recovering from a security incident and for forensics purposes. With backups in place, you can restore your systems to their previous safe state. On AWS, you can take snapshots of various resources. Snapshots provide you with point-in-time backups of those resources. There are many AWS services that can support you in backup and recovery. For detail on these services and approaches for backup and recovery, see Backup and Recovery Prescriptive Guidance and Use backups to recover from security incidents.

Especially when it comes to situations such as ransomware, it's critical for your backups to be well protected. For guidance on securing your backups, see Top 10 security best practices for securing backups in AWS. In addition to securing your backups, you should regularly test your backup and restore processes to verify that the technology and processes you have in place work as expected.

Automate forensics

During a security event, your incident response team must be able to collect and analyze evidence quickly while maintaining accuracy for the time period surrounding the event (such as capturing logs related to a specific event or resource or collecting memory dump of an Amazon EC2 instance). It's both challenging and time consuming for the incident response team to manually collect the relevant evidence, especially across a large number of instances and accounts. Additionally, manual collection can be prone to human error. For these reasons, you should develop and implement automation for forensics as much as possible.

AWS offers a number of automation resources for forensics, which are listed in the following Resources section. These resources are examples of forensics patterns that we have developed and customers have implemented. While they might be a useful reference architecture to start with, consider modifying them or creating new forensics automation patterns based on your environment, requirements, tools, and forensics processes.

Resources

Related documents:

- AWS Security Incident Response Guide - Develop Forensics Capabilities
- AWS Security Incident Response Guide - Forensics Resources
- Forensic investigation environment strategies in the AWS Cloud
- How to automate forensic disk collection in AWS
- AWS Prescriptive Guidance - Automate incident response and forensics

Related videos:

- Automating Incident Response and Forensics

Related examples:

- Automated Incident Response and Forensics Framework
- Automated Forensics Orchestrator for Amazon EC2

SEC10-BP04 Develop and test security incident response playbooks

A key part of preparing your incident response processes is developing playbooks. Incident response playbooks provide prescriptive guidance and steps to follow when a security event occurs. Having clear structure and steps simplifies the response and reduces the likelihood for human error.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Playbooks should be created for incident scenarios such as:

- **Expected incidents**: Playbooks should be created for incidents you anticipate. This includes threats like denial of service (DoS), ransomware, and credential compromise.
- **Known security findings or alerts**: Playbooks should be created to address your known security findings and alerts, such as those from Amazon GuardDuty. When you receive a GuardDuty finding, the playbook should provide clear steps to prevent mishandling or ignoring the alert. For more remediation details and guidance, see Remediating security issues discovered by GuardDuty.

Playbooks should contain technical steps for a security analyst to complete in order to adequately investigate and respond to a potential security incident.

AWS' Customer Incident Response Team (CIRT) has published a GitHub repository containing incident response playbooks, organized by threat scenario, type, and resource. These playbooks can be adapted to align with your existing incident response procedures or serve as a foundation for developing new ones.

Implementation steps

Items to include in a playbook include:

- **Playbook overview**: What risk or incident scenario does this playbook address? What is the goal of the playbook?
- **Prerequisites**: What logs, detection mechanisms, and automated tools are required for this incident scenario? What is the expected notification?
- **Communication and escalation information**: Who is involved and what is their contact information? What are each of the stakeholders' responsibilities?
- **Response steps**: Across phases of incident response, what tactical steps should be taken? What queries should an analyst run? What code should be run to achieve the desired outcome?
  - **Detect**: How will the incident be detected?
  - **Analyze**: How will the scope of impact be determined?
  - **Contain**: How will the incident be isolated to limit scope?
  - **Eradicate**: How will the threat be removed from the environment?
  - **Recover**: How will the affected system or resource be brought back into production?
- **Expected outcomes**: After queries and code are run, what is the expected result of the playbook?

Resources

Related Well-Architected best practices:

- SEC10-BP02 - Develop incident management plans

Related documents:

- Framework for Incident Response Playbooks
- Develop your own Incident Response Playbooks
- Incident Response Playbook Samples
- Building an AWS incident response runbook using Jupyter playbooks and CloudTrail Lake

SEC10-BP05 Pre-provision access

Verify that incident responders have the correct access pre-provisioned in AWS to reduce the time needed for investigation through to recovery.

Common anti-patterns:

- Using the root account for incident response.
- Altering existing accounts.
- Manipulating IAM permissions directly when providing just-in-time privilege elevation.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS recommends reducing or eliminating reliance on long-lived credentials wherever possible, in favor of temporary credentials and *just-in-time* privilege escalation mechanisms. Long-lived credentials are prone to security risk and increase operational overhead. For most management tasks, as well as incident response tasks, we recommend you implement identity federation alongside temporary escalation for administrative access. In this model, a user requests elevation to a higher level of privilege (such as an incident response role) and, provided the user is eligible for elevation, a request is sent to an approver. If the request is approved, the user receives a set of temporary AWS credentials which can be used to complete their tasks. After these credentials expire, the user must submit a new elevation request.

We recommend the use of temporary privilege escalation in the majority of incident response scenarios. The correct way to do this is to use the AWS Security Token Service and session policies to scope access.

There are scenarios where federated identities are unavailable, such as:

- Outage related to a compromised identity provider (IdP).
- Misconfiguration or human error causing broken federated access management system.
- Malicious activity such as a distributed denial of service (DDoS) event or rendering unavailability of the system.

In the preceding cases, there should be emergency *break glass* access configured to allow investigation and timely remediation of incidents. We recommend that you use a user, group,

or role with appropriate permissions to perform tasks and access AWS resources. Use the root user only for tasks that require root user credentials. To verify that incident responders have the correct level of access to AWS and other relevant systems, we recommend the pre-provisioning of dedicated accounts. The accounts require privileged access, and must be tightly controlled and monitored. The accounts must be built with the fewest privileges required to perform the necessary tasks, and the level of access should be based on the playbooks created as part of the incident management plan.

Use purpose-built and dedicated users and roles as a best practice. Temporarily escalating user or role access through the addition of IAM policies both makes it unclear what access users had during the incident, and risks the escalated privileges not being revoked.

It is important to remove as many dependencies as possible to verify that access can be gained under the widest possible number of failure scenarios. To support this, create a playbook to verify that incident response users are created as users in a dedicated security account, and not managed through any existing Federation or single sign-on (SSO) solution. Each individual responder must have their own named account. The account configuration must enforce strong password policy and multi-factor authentication (MFA). If the incident response playbooks only require access to the AWS Management Console, the user should not have access keys configured and should be explicitly disallowed from creating access keys. This can be configured with IAM policies or service control policies (SCPs) as mentioned in the AWS Security Best Practices for AWS Organizations SCPs. The users should have no privileges other than the ability to assume incident response roles in other accounts.

During an incident it might be necessary to grant access to other internal or external individuals to support investigation, remediation, or recovery activities. In this case, use the playbook mechanism mentioned previously, and there must be a process to verify that any additional access is revoked immediately after the incident is complete.

To verify that the use of incident response roles can be properly monitored and audited, it is essential that the IAM accounts created for this purpose are not shared between individuals, and that the AWS account root user is not used unless required for a specific task. If the root user is required (for example, IAM access to a specific account is unavailable), use a separate process with a playbook available to verify availability of the root user sign-in credentials and MFA token.

To configure the IAM policies for the incident response roles, consider using IAM Access Analyzer to generate policies based on AWS CloudTrail logs. To do this, grant administrator access to the incident response role on a non-production account and run through your playbooks. Once complete, a policy can be created that allows only the actions taken. This policy can then be

applied to all the incident response roles across all accounts. You might wish to create a separate IAM policy for each playbook to allow easier management and auditing. Example playbooks could include response plans for ransomware, data breaches, loss of production access, and other scenarios.

Use the incident response accounts to assume dedicated incident response IAM roles in other AWS accounts. These roles must be configured to only be assumable by users in the security account, and the trust relationship must require that the calling principal has authenticated using MFA. The roles must use tightly-scoped IAM policies to control access. Ensure that all `AssumeRole` requests for these roles are logged in CloudTrail and alerted on, and that any actions taken using these roles are logged.

It is strongly recommended that both the IAM accounts and the IAM roles are clearly named to allow them to be easily found in CloudTrail logs. An example of this would be to name the IAM accounts *<USER_ID>*-BREAK-GLASS and the IAM roles BREAK-GLASS-ROLE.

CloudTrail is used to log API activity in your AWS accounts and should be used to configure alerts on usage of the incident response roles. Refer to the blog post on configuring alerts when root keys are used. The instructions can be modified to configure the Amazon CloudWatch metric filter-to-filter on `AssumeRole` events related to the incident response IAM role:

```
{ $.eventName = "AssumeRole" && $.requestParameters.roleArn =
 "<INCIDENT_RESPONSE_ROLE_ARN>" && $.userIdentity.invokedBy NOT EXISTS && $.eventType !
= "AwsServiceEvent" }
```

As the incident response roles are likely to have a high level of access, it is important that these alerts go to a wide group and are acted upon promptly.

During an incident, it is possible that a responder might require access to systems which are not directly secured by IAM. These could include Amazon Elastic Compute Cloud instances, Amazon Relational Database Service databases, or software-as-a-service (SaaS) platforms. It is strongly recommended that rather than using native protocols such as SSH or RDP, AWS Systems Manager Session Manager is used for all administrative access to Amazon EC2 instances. This access can be controlled using IAM, which is secure and audited. It might also be possible to automate parts of your playbooks using AWS Systems Manager Run Command documents, which can reduce user error and improve time to recovery. For access to databases and third-party tools, we recommend storing access credentials in AWS Secrets Manager and granting access to the incident responder roles.

Finally, the management of the incident response IAM accounts should be added to your Joiners, Movers, and Leavers processes and reviewed and tested periodically to verify that only the intended access is allowed.

Resources

Related documents:

- Managing temporary elevated access to your AWS environment
- AWS Security Incident Response Guide
- AWS Elastic Disaster Recovery
- AWS Systems Manager Incident Manager
- Setting an account password policy for IAM users
- Using multi-factor authentication (MFA) in AWS
- Configuring Cross-Account Access with MFA
- Using IAM Access Analyzer to generate IAM policies
- Best Practices for AWS Organizations Service Control Policies in a Multi-Account Environment
- How to Receive Notifications When Your AWS Account's Root Access Keys Are Used
- Create fine-grained session permissions using IAM managed policies
- Break glass access

Related videos:

- Automating Incident Response and Forensics in AWS
- DIY guide to runbooks, incident reports, and incident response
- Prepare for and respond to security incidents in your AWS environment

SEC10-BP06 Pre-deploy tools

Verify that security personnel have the right tools pre-deployed to reduce the time for investigation through to recovery.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To automate security response and operations functions, you can use a comprehensive set of APIs and tools from AWS. You can fully automate identity management, network security, data protection, and monitoring capabilities and deliver them using popular software development methods that you already have in place. When you build security automation, your system can monitor, review, and initiate a response, rather than having people monitor your security position and manually react to events.

If your incident response teams continue to respond to alerts in the same way, they risk alert fatigue. Over time, the team can become desensitized to alerts and can either make mistakes handling ordinary situations or miss unusual alerts. Automation helps avoid alert fatigue by using functions that process the repetitive and ordinary alerts, leaving humans to handle the sensitive and unique incidents. Integrating anomaly detection systems, such as Amazon GuardDuty, AWS CloudTrail Insights, and Amazon CloudWatch Anomaly Detection, can reduce the burden of common threshold-based alerts.

You can improve manual processes by programmatically automating steps in the process. After you define the remediation pattern to an event, you can decompose that pattern into actionable logic, and write the code to perform that logic. Responders can then run that code to remediate the issue. Over time, you can automate more and more steps, and ultimately automatically handle whole classes of common incidents.

During a security investigation, you need to be able to review relevant logs to record and understand the full scope and timeline of the incident. Logs are also required for alert generation, indicating certain actions of interest have happened. It is critical to select, enable, store, and set up querying and retrieval mechanisms, and set up alerting. Additionally, an effective way to provide tools to search log data is Amazon Detective.

AWS offers over 200 cloud services and thousands of features. We recommend that you review the services that can support and simplify your incident response strategy.

In addition to logging, you should develop and implement a tagging strategy. Tagging can help provide context around the purpose of an AWS resource. Tagging can also be used for automation.

Implementation steps

Select and set up logs for analysis and alerting

See the following documentation on configuring logging for incident response:

- Logging strategies for security incident response
- SEC04-BP01 Configure service and application logging

Enable security services to support detection and response

AWS provides detective, preventative, and responsive capabilities, and other services can be used to architect custom security solutions. For a list of the most relevant services for security incident response, see Cloud capability definitions and the Security Incident Response home page.

Develop and implement a tagging strategy

Obtaining contextual information on the business use case and relevant internal stakeholders surrounding an AWS resource can be difficult. One way to do this is in the form of tags, which assign metadata to your AWS resources and consist of a user-defined key and value. You can create tags to categorize resources by purpose, owner, environment, type of data processed, and other criteria of your choice.

Having a consistent tagging strategy can speed up response times and minimize time spent on organizational context by allowing you to quickly identify and discern contextual information about an AWS resource. Tags can also serve as a mechanism to initiate response automations. For more detail on what to tag, see Tagging your AWS resources. You'll want to first define the tags you want to implement across your organization. After that, you'll implement and enforce this tagging strategy. For more detail on implementation and enforcement, see Implement AWS resource tagging strategy using AWS Tag Policies and Service Control Policies (SCPs).

Resources

Related Well-Architected best practices:

- SEC04-BP01 Configure service and application logging
- SEC04-BP02 Capture logs, findings, and metrics in standardized locations

Related documents:

- Logging strategies for security incident response
- Incident response cloud capability definitions

Related examples:

- Threat Detection and Response with Amazon GuardDuty and Amazon Detective
- Security Hub Workshop
- Vulnerability Management with Amazon Inspector

SEC10-BP07 Run simulations

As organizations grow and evolve over time, so does the threat landscape, making it important to continually review your incident response capabilities. Running simulations (also known as game days) is one method that can be used to perform this assessment. Simulations use real-world security event scenarios designed to mimic a threat actor's tactics, techniques, and procedures (TTPs) and allow an organization to exercise and evaluate their incident response capabilities by responding to these mock cyber events as they might occur in reality.

Benefits of establishing this best practice: Simulations have a variety of benefits:

- Validating cyber readiness and developing the confidence of your incident responders.
- Testing the accuracy and efficiency of tools and workflows.
- Refining communication and escalation methods aligned with your incident response plan.
- Providing an opportunity to respond to less common vectors.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

There are three main types of simulations:

- **Tabletop exercises:** The tabletop approach to simulations is a discussion-based session involving the various incident response stakeholders to practice roles and responsibilities and use established communication tools and playbooks. Exercise facilitation can typically be accomplished in a full day in a virtual venue, physical venue, or a combination. Because it is discussion-based, the tabletop exercise focuses on processes, people, and collaboration. Technology is an integral part of the discussion, but the actual use of incident response tools or scripts is generally not a part of the tabletop exercise.
- **Purple team exercises:** Purple team exercises increase the level of collaboration between the incident responders (blue team) and simulated threat actors (red team). The blue team is comprised of members of the security operations center (SOC), but can also include other stakeholders that would be involved during an actual cyber event. The red team is comprised

of a penetration testing team or key stakeholders that are trained in offensive security. The red team works collaboratively with the exercise facilitators when designing a scenario so that the scenario is accurate and feasible. During purple team exercises, the primary focus is on the detection mechanisms, the tools, and the standard operating procedures (SOPs) supporting the incident response efforts.

- **Red team exercises:** During a red team exercise, the offense (red team) conducts a simulation to achieve a certain objective or set of objectives from a predetermined scope. The defenders (blue team) will not necessarily have knowledge of the scope and duration of the exercise, which provides a more realistic assessment of how they would respond to an actual incident. Because red team exercises can be invasive tests, be cautious and implement controls to verify that the exercise does not cause actual harm to your environment.

Consider facilitating cyber simulations at a regular interval. Each exercise type can provide unique benefits to the participants and the organization as a whole, so you might choose to start with less complex simulation types (such as tabletop exercises) and progress to more complex simulation types (red team exercises). You should select a simulation type based on your security maturity, resources, and your desired outcomes. Some customers might not choose to perform red team exercises due to complexity and cost.

Implementation steps

Regardless of the type of simulation you choose, simulations generally follow these implementation steps:

1. **Define core exercise elements:** Define the simulation scenario and the objectives of the simulation. Both of these should have leadership acceptance.
2. **Identify key stakeholders:** At a minimum, an exercise needs exercise facilitators and participants. Depending on the scenario, additional stakeholders such as legal, communications, or executive leadership might be involved.
3. **Build and test the scenario:** The scenario might need to be redefined as it is being built if specific elements aren't feasible. A finalized scenario is expected as the output of this stage.
4. **Facilitate the simulation:** The type of simulation determines the facilitation used (a paper-based scenario compared to a highly technical, simulated scenario). The facilitators should align their facilitation tactics to the exercise objects and they should engage all exercise participants wherever possible to provide the most benefit.

5. **Develop the after-action report (AAR):** Identify areas that went well, those that can use improvement, and potential gaps. The AAR should measure the effectiveness of the simulation as well as the team's response to the simulated event so that progress can be tracked over time with future simulations.

Resources

Related documents:

- AWS Incident Response Guide

Related videos:

- AWS GameDay - Security Edition
- Running effective security incident response simulations

SEC10-BP08 Establish a framework for learning from incidents

Implementing a *lessons learned* framework and root cause analysis capability can not only help improve incident response capabilities, but also help prevent the incident from recurring. By learning from each incident, you can help avoid repeating the same mistakes, exposures, or misconfigurations, not only improving your security posture, but also minimizing time lost to preventable situations.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

It's important to implement a *lessons learned* framework that establishes and achieves, at a high level, the following points:

- When is a lessons learned held?
- What is involved in the lessons learned process?
- How is a lessons learned performed?
- Who is involved in the process and how?
- How will areas of improvement be identified?
- How will you ensure improvements are effectively tracked and implemented?

The framework should not focus on or blame individuals, but instead should focus on improving tools and processes.

Implementation steps

Aside from the preceding high-level outcomes listed, it's important to make sure that you ask the right questions to derive the most value (information that leads to actionable improvements) from the process. Consider these questions to help get you started in fostering your lessons learned discussions:

- What was the incident?
- When was the incident first identified?
- How was it identified?
- What systems alerted on the activity?
- What systems, services, and data were involved?
- What specifically occurred?
- What worked well?
- What didn't work well?
- Which process or procedures failed or failed to scale to respond to the incident?
- What can be improved within the following areas:
  - **People**
    - Were the people who were needed to be contacted actually available and was the contact list up to date?
    - Were people missing training or capabilities needed to effectively respond and investigate the incident?
    - Were the appropriate resources ready and available?
  - **Process**
    - Were processes and procedures followed?
    - Were processes and procedures documented and available for this (type of) incident?
    - Were required processes and procedures missing?
    - Were the responders able to gain timely access to the required information to respond to the issue?
  - **Technology**
    - Did existing alerting systems effectively identify and alert on the activity?

- How could we have reduced time-to-detection by 50%?
- Do existing alerts need improvement or new alerts need to be built for this (type of) incident?
- Did existing tools allow for effective investigation (search/analysis) of the incident?
- What can be done to help identify this (type of) incident sooner?
- What can be done to help prevent this (type of) incident from occurring again?
- Who owns the improvement plan and how will you test that it has been implemented?
- What is the timeline for the additional monitoring or preventative controls and processes to be implemented and tested?

This list isn't all-inclusive, but is intended to serve as a starting point for identifying what the organization and business needs are and how you can analyze them in order to most effectively learn from incidents and continuously improve your security posture. Most important is getting started by incorporating lessons learned as a standard part of your incident response process, documentation, and expectations across the stakeholders.

Resources

Related documents:

- AWS Security Incident Response Guide - Establish a framework for learning from incidents
- NCSC CAF guidance - Lessons learned

Application security

Question

- SEC 11. How do you incorporate and validate the security properties of applications throughout the design, development, and deployment lifecycle?

SEC 11. How do you incorporate and validate the security properties of applications throughout the design, development, and deployment lifecycle?

Training people, testing using automation, understanding dependencies, and validating the security properties of tools and applications help to reduce the likelihood of security issues in production workloads.

Best practices

- SEC11-BP01 Train for application security
- SEC11-BP02 Automate testing throughout the development and release lifecycle
- SEC11-BP03 Perform regular penetration testing
- SEC11-BP04 Conduct code reviews
- SEC11-BP05 Centralize services for packages and dependencies
- SEC11-BP06 Deploy software programmatically
- SEC11-BP07 Regularly assess security properties of the pipelines
- SEC11-BP08 Build a program that embeds security ownership in workload teams

SEC11-BP01 Train for application security

Provide training to your team on secure development and operation practices, which helps them build secure and high-quality software. This practice helps your team to prevent, detect, and remediate security issues earlier in the development lifecycle. Consider training that covers threat modeling, secure coding practices, and using services for secure configurations and operations. Provide your team access to training through self-service resources, and regularly gather their feedback for continuous improvement.

Desired outcome: You equip your team with the knowledge and skills necessary to design and build software with security in mind from the outset. Through training on threat modeling and secure development practices, your team has a deep understanding of potential security risks and how to mitigate them during the software development lifecycle (SDLC). This proactive approach to security is part of your team's culture, and you become able to identify and remediate potential security issues early on. As a result, your team delivers high-quality, secure software and features more efficiently, which accelerates the overall delivery timeline. You have a collaborative and inclusive security culture within your organization, where the ownership of security is shared across all builders.

Common anti-patterns:

- You wait until a security review, and then consider the security properties of a system.
- You leave all security decisions to a central security team.
- You don't communicate how the decisions taken in the SDLC relate to the overall security expectations or policies of the organization.

- You perform the security review process too late.

Benefits of establishing this best practice:

- Better knowledge of the organizational requirements for security early in the development cycle.
- Being able to identify and remediate potential security issues faster, resulting in a quicker delivery of features.
- Improved quality of software and systems.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To build secure and high-quality software, provide training to your team on common practices for secure development and operation of applications. This practice can help your team prevent, detect, and remediate security issues earlier in the development lifecycle, which can accelerate your delivery timeline.

To achieve this practice, consider training your team on threat modeling using AWS resources like the Threat Modeling Workshop. Threat modeling can help your team understand potential security risks and design systems with security in mind from the outset. Additionally, you can provide access to AWS Training and Certification, industry, or AWS Partner training on secure development practices. For more detail on a comprehensive approach to designing, developing, securing, and efficiently operating at scale, see AWS DevOps Guidance.

Clearly define and communicate your organization's security review process, and outline the responsibilities of your team, the security team, and other stakeholders. Publish self-service guidance, code examples, and templates that demonstrate how to meet your security requirements. You can use AWS services like AWS CloudFormation, AWS Cloud Development Kit (AWS CDK) (AWS CDK) Constructs, and Service Catalog to provide pre-approved, secure configurations and reduce the need for custom setups.

Regularly gather feedback from your team on their experience with the security review process and training, and use this feedback to continuously improve. Conduct game days or bug bash campaigns to identify and address security issues while simultaneously enhancing your team's skills.

Implementation steps

1. **Identify training needs**: Assess the current skill level and knowledge gaps within your team regarding secure development practices through surveys, code reviews, or discussions with team members.
2. **Plan the training**: Based on the identified needs, create a training plan that covers relevant topics such as threat modeling, secure coding practices, security testing, and secure deployment practices. Employ resources like the Threat Modeling Workshop, AWS Training and Certification, and industry or AWS Partner training programs.
3. **Schedule and deliver training**: Schedule regular training sessions or workshops for your team. These can be instructor-led or self-paced, depending on your team's preferences and availability. Encourage hands-on exercises and practical examples to reinforce the learning.
4. **Define a security review process**: Collaborate with your security team and other stakeholders to clearly define the security review process for your applications. Document the responsibilities of each team or individual involved in the process, including your development team, security team, and other relevant stakeholders.
5. **Create self-service resources**: Develop self-service guidance, code examples, and templates that demonstrate how to meet your organization's security requirements. Consider AWS services like CloudFormation, AWS CDK Constructs, and Service Catalog to provide pre-approved, secure configurations and reduce the need for custom setups.
6. **Communicate and socialize**: Effectively communicate the security review process and the available self-service resources to your team. Conduct training sessions or workshops to familiarize them with these resources, and verify that they understand how to use them.
7. **Gather feedback and improve**: Regularly collect feedback from your team on their experience with the security review process and training. Use this feedback to identify areas for improvement and continuously refine the training materials, self-service resources, and the security review process.
8. **Conduct security exercises**: Organize game days or bug bash campaigns to identify and address security issues within your applications. These exercises not only help uncover potential vulnerabilities but also serve as practical learning opportunities for your team that enhance their skills in secure development and operation.
9. **Continue to learn and improve:** Encourage your team to stay up to date with the latest secure development practices, tools, and techniques. Regularly review and update your training materials and resources to reflect the evolving security landscape and best practices.

Resources

Related best practices:

- SEC11-BP08 Build a program that embeds security ownership in workload teams

Related documents:

- AWS Training and Certification
- How to think about cloud security governance
- How to approach threat modeling
- Accelerating training – The AWS Skills Guild
- AWS DevOps Sagas

Related videos:

- Proactive security: Considerations and approaches

Related examples:

- Workshop on threat modeling
- Industry awareness for developers

Related services:

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK) (AWS CDK) Constructs
- Service Catalog

SEC11-BP02 Automate testing throughout the development and release lifecycle

Automate the testing for security properties throughout the development and release lifecycle. Automation makes it easier to consistently and repeatably identify potential issues in software prior to release, which reduces the risk of security issues in the software being provided.

Desired outcome: The goal of automated testing is to provide a programmatic way of detecting potential issues early and often throughout the development lifecycle. When you automate regression testing, you can rerun functional and non-functional tests to verify that previously tested software still performs as expected after a change. When you define security unit tests to check for common misconfigurations, such as broken or missing authentication, you can identify and fix these issues early in the development process.

Test automation uses purpose-built test cases for application validation, based on the application's requirements and desired functionality. The result of the automated testing is based on comparing the generated test output to its respective expected output, which expedites the overall testing lifecycle. Testing methodologies such as regression testing and unit test suites are best suited for automation. Automating the testing of security properties allows builders to receive automated feedback without having to wait for a security review. Automated tests in the form of static or dynamic code analysis can increase code quality and help detect potential software issues early in the development lifecycle.

Common anti-patterns:

- Not communicating the test cases and test results of the automated testing.
- Performing the automated testing only immediately prior to a release.
- Automating test cases with frequently changing requirements.
- Failing to provide guidance on how to address the results of security tests.

Benefits of establishing this best practice:

- Reduced dependency on people evaluating the security properties of systems.
- Having consistent findings across multiple workstreams improves consistency.
- Reduced likelihood of introducing security issues into production software.
- Shorter window of time between detection and remediation due to catching software issues earlier.
- Increased visibility of systemic or repeated behavior across multiple workstreams, which can be used to drive organization-wide improvements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As you build your software, adopt various mechanisms for software testing to ensure that you are testing your application for both functional requirements, based on your application's business logic, and non-functional requirements, which are focused on application reliability, performance, and security.

Static application security testing (SAST) analyzes your source code for anomalous security patterns, and provides indications for defect prone code. SAST relies on static inputs, such as documentation (requirements specification, design documentation, and design specifications) and application source code to test for a range of known security issues. Static code analyzers can help expedite the analysis of large volumes of code. The NIST Quality Group provides a comparison of Source Code Security Analyzers, which includes open source tools for Byte Code Scanners and Binary Code Scanners.

Complement your static testing with dynamic analysis security testing (DAST) methodologies, which performs tests against the running application to identify potentially unexpected behavior. Dynamic testing can be used to detect potential issues that are not detectable via static analysis. Testing at the code repository, build, and pipeline stages allows you to check for different types of potential issues from entering into your code. Amazon Q Developers provides code recommendations, including security scanning, in the builder's IDE. Amazon CodeGuru Security can identify critical issues, security issues, and hard-to-find bugs during application development, and provides recommendations to improve code quality. Extracting Software Bill of Materials (SBOM) also allows you to extract a formal record containing the details and relationships of the various components used in building your software. This allows you to inform vulnerability management, and quickly identify software or component dependencies and supply chain risks.

The Security for Developers workshop uses AWS developer tools, such as AWS CodeBuild, AWS CodeCommit, and AWS CodePipeline, for release pipeline automation that includes SAST and DAST testing methodologies.

As you progress through your SDLC, establish an iterative process that includes periodic application reviews with your security team. Feedback gathered from these security reviews should be addressed and validated as part of your release readiness review. These reviews establish a robust application security posture, and provide builders with actionable feedback to address potential issues.

Implementation steps

- Implement consistent IDE, code review, and CI/CD tools that include security testing.

- Consider where in the SDLC it is appropriate to block pipelines instead of just notifying builders that issues need to be remediated.
- Automated Security Helper (ASH) is an example for open-source code security scanning tool.
- Performing testing or code analysis using automated tools, such as Amazon Q Developer integrated with developer IDEs, and Amazon CodeGuru Security for scanning code on commit, helps builders get feedback at the right time.
- When building using AWS Lambda, you can use Amazon Inspector to scan the application code in your functions.
- When automated testing is included in CI/CD pipelines, you should use a ticketing system to track the notification and remediation of software issues.
- For security tests that might generate findings, linking to guidance for remediation helps builders improve code quality.
- Regularly analyze the findings from automated tools to prioritize the next automation, builder training, or awareness campaign.
- To extract SBOM as part of your CI/CD pipelines, use Amazon Inspector SBOM Generator to produce SBOMs for archives, container images, directories, local systems, and compiled Go and Rust binaries in the CycloneDX SBOM format.

Resources

Related best practices:

- DevOps Guidance: DL.CR.3 Establish clear completion criteria for code tasks

Related documents:

- Continuous Delivery and Continuous Deployment

- AWS DevOps Competency Partners
- AWS Security Competency Partners for Application Security
- Choosing a Well-Architected CI/CD approach
- Secrets detection in Amazon CodeGuru Security
- Amazon CodeGuru Security Detection Library
- Accelerate deployments on AWS with effective governance

- How AWS approaches automating safe, hands-off deployments
- How Amazon CodeGuru Security helps you effectively balance security and velocity

Related videos:

- Hands-off: Automating continuous delivery pipelines at Amazon
- Automating cross-account CI/CD pipelines
- The Software Development Prcess at Amazon
- Testing software and systems at Amazon

Related examples:

- Industry awareness for developers
- Automated Security Helper (ASH)
- AWS CodePipeline Governance - Github

SEC11-BP03 Perform regular penetration testing

Perform regular penetration testing of your software. This mechanism helps identify potential software issues that cannot be detected by automated testing or a manual code review. It can also help you understand the efficacy of your detective controls. Penetration testing should try to determine if the software can be made to perform in unexpected ways, such as exposing data that should be protected, or granting broader permissions than expected.

Desired outcome: Penetration testing is used to detect, remediate, and validate your application's security properties. Regular and scheduled penetration testing should be performed as part of the software development lifecycle (SDLC). The findings from penetration tests should be addressed prior to the software being released. You should analyze the findings from penetration tests to identify if there are issues that could be found using automation. Having a regular and repeatable penetration testing process that includes an active feedback mechanism helps inform the guidance to builders and improves software quality.

Common anti-patterns:

- Only penetration testing for known or prevalent security issues.
- Penetration testing applications without dependent third-party tools and libraries.

- Only penetration testing for package security issues, and not evaluating implemented business logic.

Benefits of establishing this best practice:

- Increased confidence in the security properties of the software prior to release.
- Opportunity to identify preferred application patterns, which leads to greater software quality.
- A feedback loop that identifies earlier in the development cycle where automation or additional training can improve the security properties of software.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Penetration testing is a structured security testing exercise where you run planned security breach scenarios to detect, remediate, and validate security controls. Penetration tests start with reconnaissance, during which data is gathered based on the current design of the application and its dependencies. A curated list of security-specific testing scenarios are built and run. The key purpose of these tests is to uncover security issues in your application, which could be exploited for gaining unintended access to your environment, or unauthorized access to data. You should perform penetration testing when you launch new features, or whenever your application has undergone major changes in function or technical implementation.

You should identify the most appropriate stage in the development lifecycle to perform penetration testing. This testing should happen late enough that the functionality of the system is close to the intended release state, but with enough time remaining for any issues to be remediated.

Implementation steps

- Have a structured process for how penetration testing is scoped, basing this process on the threat model is a good way of maintaining context.
- Identify the appropriate place in the development cycle to perform penetration testing. This should be when there is minimal change expected in the application, but with enough time to perform remediation.
- Train your builders on what to expect from penetration testing findings, and how to get information on remediation.

- Use tools to speed up the penetration testing process by automating common or repeatable tests.
- Analyze penetration testing findings to identify systemic security issues, and use this data to inform additional automated testing and ongoing builder education.

Resources

Related best practices:

- SEC11-BP01 Train for application security
- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- AWS Penetration Testing provides detailed guidance for penetration testing on AWS
- Accelerate deployments on AWS with effective governance
- AWS Security Competency Partners
- Modernize your penetration testing architecture on AWS Fargate
- AWS Fault injection Simulator

Related examples:

- Automate API testing with AWS CodePipeline (GitHub)
- Automated security helper (GitHub)

SEC11-BP04 Conduct code reviews

Implement code reviews to help verify the quality and security of software being developed. Code reviews involve having team members other than the original code author review the code for potential issues, vulnerabilities, and adherence to coding standards and best practices. This process helps catch errors, inconsistencies, and security flaws that might have been overlooked by the original developer. Use automated tools to assist with code reviews.

Desired outcome: You include code reviews during development to increase the quality of the software being written. You upskill less experienced members of the team through learnings

identified during the code review. You identify opportunities for automation and support the code review process using automated tools and testing.

Common anti-patterns:

- You don't review code before deployment.
- The same person writes and reviews the code.
- You don't use automation and tools to assist or orchestrate code reviews.
- You don't train builders on application security before they review code.

Benefits of establishing this best practice:

- Increased code quality.
- Increased consistency of code development through reuse of common approaches.
- Reduction in the number of issues discovered during penetration testing and later stages.
- Improved knowledge transfer within the team.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Code reviews help to verify the quality and security of the software during development. Manual reviews involve having a team member other than the original code author review the code for potential issues, vulnerabilities, and adherence to coding standards and best practices. This process helps catch errors, inconsistencies, and security flaws that might have been overlooked by the original developer.

Consider Amazon CodeGuru Security to help conduct automated code reviews. CodeGuru Security uses machine learning and automated reasoning to analyze your code and identify potential security vulnerabilities and coding issues. Integrate automated code reviews with your existing code repositories and continuous integration/continuous deployment (CI/CD) pipelines.

Implementation steps

1. **Establish a code review process:**
   - Define when code reviews should occur, such as before merging code into the main branch or before deploying to production.

- Determine who should be involved in the code review process, such as team members, senior developers, and security experts.
- Decide on the code review methodology, including the process and tools to be used.

2. **Set up code review tools:**
   - Evaluate and select code review tools that fit your team's needs, such as GitHub Pull Requests or CodeGuru Security
   - Integrate the chosen tools with your existing code repositories and CI/CD pipelines.
   - Configure the tools to enforce code review requirements, such as the minimum number of reviewers and approval rules.
3. **Define a code review checklist and guidelines:**
   - Create a code review checklist or guidelines that outline what should be reviewed. Consider factors such as code quality, security vulnerabilities, adherence to coding standards, and performance.
   - Share the checklist or guidelines with the development team, and verify everyone understands the expectations.
4. **Train developers on code review best practices:**
   - Provide training to your team on how to conduct effective code reviews.
   - Educate your team on application security principles and common vulnerabilities to look for during reviews.
   - Encourage knowledge sharing and pair programming sessions to upskill less experienced team members.
5. **Implement the code review process:**
   - Integrate the code review step into your development workflow, such as creating a pull request and assigning reviewers.
   - Require that code changes undergo a code review before merge or deployment.
   - Encourage open communication and constructive feedback during the review process.
6. **Monitor and improve:**
   - Regularly review the effectiveness of your code review process and gather feedback from the team.
   - Identify opportunities for automation or tool improvements to streamline the code review process.

- Continuously update and refine the code review checklist or guidelines based on learnings and industry best practices.

7. **Foster a culture of code review:**
   - Emphasize the importance of code reviews to maintain code quality and security.
   - Celebrate successes and learnings from the code review process.
   - Encourage a collaborative and supportive environment where developers feel comfortable giving and receiving feedback.

Resources

Related best practices:

- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- DevOps Guidance: DL.CR.2 Perform peer review for code changes
- About pull requests in GitHub

Related examples:

- Automate code reviews with Amazon CodeGuru Security
- Automating detection of security vulnerabilities and bugs in CI/CD pipelines using Amazon CodeGuru Security CLI

Related videos:

- Continuous improvement of code quality with Amazon CodeGuru Security

SEC11-BP05 Centralize services for packages and dependencies

Provide centralized services for your teams to obtain software packages and other dependencies. This allows the validation of packages before they are included in the software that you write and provides a source of data for the analysis of the software being used in your organization.

Desired outcome: You build your workload from external software packages in addition to the code that you write. This makes it simpler for you to implement functionality that is repeatedly used, such as a JSON parser or an encryption library. You centralize the sources for these packages and dependencies so your security team can validate them before they are used. You use this approach in conjunction with the manual and automated testing flows to increase the confidence in the quality of the software that you develop.

Common anti-patterns:

- You pull packages from arbitrary repositories on the internet.
- You don't test new packages before you make them available to builders.

Benefits of establishing this best practice:

- Better understanding of what packages are being used in the software being built.
- Being able to notify workload teams when a package needs to be updated based on the understanding of who is using what.
- Reducing the risk of a package with issues being included in your software.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Provide centralized services for packages and dependencies in a way that is simple for builders to consume. Centralized services can be logically central rather than implemented as a monolithic system. This approach allows you to provide services in a way that meets the needs of your builders. You should implement an efficient way of adding packages to the repository when updates happen or new requirements emerge. AWS services such as AWS CodeArtifact or similar AWS partner solutions provide a way of delivering this capability.

Implementation steps

- Implement a logically centralized repository service that is available in all of the environments where software is developed.
- Include access to the repository as part of the AWS account vending process.
- Build automation to test packages before they are published in a repository.

- Maintain metrics of the most commonly used packages, languages, and teams with the highest amount of change.
- Provide an automated mechanism for builder teams to request new packages and provide feedback.
- Regularly scan packages in your repository to identify the potential impact of newly discovered issues.

Resources

Related best practices:

- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- DevOps Guidance: DL.CS.2 Sign code artifacts after each build
- Supply chain Levels for Software Artifacts (SLSA)

Related examples:

- Accelerate deployments on AWS with effective governance
- Tighten your package security with CodeArtifact Package Origin Control toolkit
- Multi Region Package Publishing Pipeline (GitHub)
- Publishing Node.js Modules on AWS CodeArtifact using AWS CodePipeline (GitHub)
- AWS CDK Java CodeArtifact Pipeline Sample (GitHub)
- Distribute private .NET NuGet packages with AWS CodeArtifact (GitHub)

Related videos:

- Proactive security: Considerations and approaches
- The AWS Philosophy of Security (re:Invent 2017)
- When security, safety, and urgency all matter: Handling Log4Shell

SEC11-BP06 Deploy software programmatically

Perform software deployments programmatically where possible. This approach reduces the likelihood that a deployment fails or an unexpected issue is introduced due to human error.

Desired outcome: The version of your workload that you test is the version that you deploy, and the deployment is performed consistently every time. You externalize the configuration of your workload, which helps you deploy to different environments without changes. You employ cryptographic signing of your software packages to verify that nothing changes between environments.

Common anti-patterns:

- Manually deploying software into production.
- Manually performing changes to software to cater to different environments.

Benefits of establishing this best practice:

- Increased confidence in the software release process.
- Reduced risk of a failed change impacting business functionality.
- Increased release cadence due to lower change risk.
- Automatic rollback capability for unexpected events during deployment.
- Ability to cryptographically prove that the software that was tested is the software deployed.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To maintain a robust and reliable application infrastructure, implement secure and automated deployment practices. This practice involves removing persistent human access from production environments, using CI/CD tools for deployments, and externalizing environment-specific configuration data. By following this approach, you can enhance security, reduce the risk of human errors, and streamline the deployment process.

You can build your AWS account structure to remove persistent human access from production environments. This practice minimizes the risk of unauthorized changes or accidental modifications, which improves the integrity of your production systems. Instead of direct human access, you can use CI/CD tools like AWS CodeBuild and AWS CodePipeline to perform

deployments. You can use these services to automate the build, test, and deployment processes, which reduces manual intervention and increases consistency.

To further enhance security and traceability, you can sign your application packages after they have been tested and validate these signatures during deployment. To do so, use cryptographic tools such as AWS Signer or AWS Key Management Service (AWS KMS). By signing and verifying packages, you can make sure that you deploy only authorized and validated code to your environments.

Additionally, your team can architect your workload to obtain environment-specific configuration data from an external source, such as AWS Systems Manager Parameter Store. This practice separates the application code from the configuration data, which helps you manage and update configurations independently without modifying the application code itself.

To streamline infrastructure provisioning and management, consider using infrastructure as code (IaC) tools like AWS CloudFormation or AWS CDK. You can use these tools to define your infrastructure as code, which improves the consistency and repeatability of deployments across different environments.

Consider canary deployments to validate the successful deployment of your software. Canary deployments involve rolling out changes to a subset of instances or users before deploying to the entire production environment. You can then monitor the impact of changes and roll back if necessary, which minimizes the risk of widespread issues.

Follow the recommendations outlined in the Organizing Your AWS Environment Using Multiple Accounts whitepaper. This whitepaper provides guidance on separating environments (such as development, staging, and production) into distinct AWS accounts, which further enhances security and isolation.

Implementation steps

1. **Set up AWS account structure**:
   - Follow the guidance in the Organizing Your AWS Environment Using Multiple Accounts whitepaper to create separate AWS accounts for different environments (for exampoe, development, staging, and production).
   - Configure appropriate access controls and permissions for each account to restrict direct human access to production environments.
2. **Implement a CI/CD pipeline**:
   - Set up a CI/CD pipeline using services like AWS CodeBuild and AWS CodePipeline.

- Configure the pipeline to automatically build, test, and deploy your application code to the respective environments.
- Integrate code repositories with the CI/CD pipeline for version control and code management.

3. **Sign and verify application packages**:
   - Use AWS Signer or AWS Key Management Service (AWS KMS) to sign your application packages after they have been tested and validated.
   - Configure the deployment process to verify the signatures of the application packages before you deploy them to the target environments.
4. **Externalize configuration data**:
   - Store environment-specific configuration data in AWS Systems Manager Parameter Store.
   - Modify your application code to retrieve configuration data from the Parameter Store during deployment or runtime.
5. **Implement infrastructure as code (IaC)**:
   - Use IaC tools like AWS CloudFormation or AWS CDK to define and manage your infrastructure as code.
   - Create CloudFormation templates or CDK scripts to provision and configure the necessary AWS resources for your application.
   - Integrate IaC with your CI/CD pipeline to automatically deploy infrastructure changes alongside application code changes.
6. **Implement canary deployments**:
   - Configure your deployment process to support canary deployments, where changes are rolled out to a subset of instances or users before you deploy them to the entire production environment.
   - Use services like AWS CodeDeploy or AWS ECS to manage canary deployments and monitor the impact of changes.
   - Implement rollback mechanisms to revert to the previous stable version if issues are detected during the canary deployment.
7. **Monitor and audit**:
   - Set up monitoring and logging mechanisms to track deployments, application performance, and infrastructure changes.
   - Use services like Amazon CloudWatch and AWS CloudTrail to collect and analyze logs and metrics.

- Implement auditing and compliance checks to verify adherence to security best practices and regulatory requirements.

8. **Continuously improve:**
   - Regularly review and update your deployment practices, and incorporate feedback and lessons learned from previous deployments.
   - Automate as much of the deployment process as possible to reduce manual intervention and potential human errors.
   - Collaborate with cross-functional teams (for example, operations or security) to align and continuously improve deployment practices.

By following these steps, you can implement secure and automated deployment practices in your AWS environment, which enhances security, reduces the risk of human errors, and streamlines the deployment process.

Resources

Related best practices:

- SEC11-BP02 Automate testing throughout the development and release lifecycle
- DL.CI.2 Trigger builds automatically upon source code modifications

Related documents:

- Accelerate deployments on AWS with effective governance
- Automating safe, hands-off deployments
- Code signing using AWS Certificate Manager Private CA and AWS Key Management Service asymmetric keys
- Code Signing, a Trust and Integrity Control for AWS Lambda

Related videos:

- Hands-off: Automating continuous delivery pipelines at Amazon

Related examples:

- Blue/Green deployments with AWS Fargate

SEC11-BP07 Regularly assess security properties of the pipelines

Apply the principles of the Well-Architected Security Pillar to your pipelines, with particular attention to the separation of permissions. Regularly assess the security properties of your pipeline infrastructure. Effectively managing the security *of* the pipelines allows you to deliver the security of the software that passes *through* the pipelines.

Desired outcome: The pipelines you use to build and deploy your software follow the same recommended practices as any other workload in your environment. The tests that you implement in your pipelines are not editable by the teams who use them. You give the pipelines only the permissions needed for the deployments they are doing using temporary credentials. You implement safeguards to prevent pipelines from deploying to the wrong environments. You configure your pipelines to emit state so that the integrity of your build environments can be validated.

Common anti-patterns:

- Security tests that can be bypassed by builders.
- Overly broad permissions for deployment pipelines.
- Pipelines not being configured to validate inputs.
- Not regularly reviewing the permissions associated with your CI/CD infrastructure.
- Use of long-term or hardcoded credentials.

Benefits of establishing this best practice:

- Greater confidence in the integrity of the software that is built and deployed through the pipelines.
- Ability to stop a deployment when there is suspicious activity.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Your deployment pipelines are a critical component of your software development lifecycle and should follow the same security principles and practices as any other workload in your environment. This includes implementing proper access controls, validating inputs, and regularly reviewing and auditing the permissions associated with your CI/CD infrastructure.

Verify that the teams responsible for building and deploying applications do not have the ability to edit or bypass the security tests and checks implemented in your pipelines. This separation of concerns helps maintain the integrity of your build and deployment processes.

As a starting point, consider employing the AWS Deployment Pipelines Reference Architecture. This reference architecture provides a secure and scalable foundation for building your CI/CD pipelines on AWS.

Additionally, you can use services like AWS Identity and Access Management Access Analyzer to generate least-privilege IAM policies for both your pipeline permissions and as a step in your pipeline to verify workload permissions. This helps verify that your pipelines and workloads have only the necessary permissions required for their specific functions, which reduces the risk of unauthorized access or actions.

Implementation steps

- Start with the AWS Deployment Pipelines Reference Architecture.
- Consider using AWS IAM Access Analyzer to programmatically generate least privilege IAM policies for the pipelines.
- Integrate your pipelines with monitoring and alerting so that you are notified of unexpected or abnormal activity, for AWS managed services Amazon EventBridge allows you to route data to targets such as AWS Lambda or Amazon Simple Notification Service (Amazon SNS).

Resources

Related documents:

- AWS Deployment Pipelines Reference Architecture
- Monitoring AWS CodePipeline
- Security best practices for AWS CodePipeline

Related examples:

- DevOps monitoring dashboard (GitHub)

SEC11-BP08 Build a program that embeds security ownership in workload teams

Build a program or mechanism that empowers builder teams to make security decisions about the software that they create. Your security team still needs to validate these decisions during a review, but embedding security ownership in builder teams allows for faster, more secure workloads to be built. This mechanism also promotes a culture of ownership that positively impacts the operation of the systems you build.

Desired outcome: You have embedded security ownership and decision-making in your teams. You have either trained your teams on how to think about security or have augmented their team with embedded or associated security people. Your teams make higher-quality security decisions earlier in the development cycle as a result.

Common anti-patterns:

- Leaving all security design decisions to a security team.
- Not addressing security requirements early enough in the development process.
- Not obtaining feedback from builders and security people on the operation of the program.

Benefits of establishing this best practice:

- Reduced time to complete security reviews.
- Reduction in security issues that are only detected at the security review stage.
- Improvement in the overall quality of the software being written.
- Opportunity to identify and understand systemic issues or areas of high value improvement.
- Reduction in the amount of rework required due to security review findings.
- Improvement in the perception of the security function.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Start with the guidance in SEC11-BP01 Train for application security. Then identify the operational model for the program that you think might work best for your organization. The two main patterns are to train builders or to embed security people in builder teams. After you have decided on the initial approach, you should pilot with a single or small group of workload teams to prove

the model works for your organization. Leadership support from the builder and security parts of the organization helps with the delivery and success of the program. As you build this program, it's important to choose metrics that can be used to show the value of the program. Learning from how AWS has approached this problem is a good learning experience. This best practice is very much focused on organizational change and culture. The tools that you use should support the collaboration between the builder and security communities.

Implementation steps

- Start by training your builders for application security.
- Create a community and an onboarding program to educate builders.
- Pick a name for the program. Guardians, Champions, or Advocates are commonly used.
- Identify the model to use: train builders, embed security engineers, or have affinity security roles.
- Identify project sponsors from security, builders, and potentially other relevant groups.
- Track metrics for the number of people involved in the program, the time taken for reviews, and the feedback from builders and security people. Use these metrics to make improvements.

Resources

Related best practices:

- SEC11-BP01 Train for application security
- SEC11-BP02 Automate testing throughout the development and release lifecycle

Related documents:

- How to approach threat modeling
- How to think about cloud security governance
- How AWS built the Security Guardians program, a mechanism to distribute security ownership
- How to build a Security Guardians program to distribute security ownership

Related videos:

- Proactive security: Considerations and approaches
- AppSec tooling and culture tips from AWS and Toyota Motor North America

Reliability

The Reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. You can find prescriptive guidance on implementation in the Reliability Pillar whitepaper.

Best practice areas

- Foundations
- Workload architecture
- Change management
- Failure management

Foundations

Questions

- REL 1. How do you manage Service Quotas and constraints?
- REL 2. How do you plan your network topology?

REL 1. How do you manage Service Quotas and constraints?

For cloud-based workload architectures, there are Service Quotas (which are also referred to as service limits). These quotas exist to prevent accidentally provisioning more resources than you need and to limit request rates on API operations so as to protect services from abuse. There are also resource constraints, for example, the rate that you can push bits down a fiber-optic cable, or the amount of storage on a physical disk.

Best practices

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover

REL01-BP01 Aware of service quotas and constraints

Be aware of your default quotas and manage your quota increase requests for your workload architecture. Know which cloud resource constraints, such as disk or network, are potentially impactful.

Desired outcome: Customers can prevent service degradation or disruption in their AWS accounts by implementing proper guidelines for monitoring key metrics, infrastructure reviews, and automation remediation steps to verify that services quotas and constraints are not reached that could cause service degradation or disruption.

Common anti-patterns:

- Deploying a workload without understanding the hard or soft quotas and their limits for the services used.
- Deploying a replacement workload without analyzing and reconfiguring the necessary quotas or contacting Support in advance.
- Assuming that cloud services have no limits and the services can be used without consideration to rates, limits, counts, quantities.
- Assuming that quotas will automatically be increased.
- Not knowing the process and timeline of quota requests.
- Assuming that the default cloud service quota is the identical for every service compared across regions.
- Assuming that service constraints can be breached and the systems will auto-scale or add increase the limit beyond the resource's constraints
- Not testing the application at peak traffic in order to stress the utilization of its resources.
- Provisioning the resource without analysis of the required resource size.
- Overprovisioning capacity by choosing resource types that go well beyond actual need or expected peaks.
- Not assessing capacity requirements for new levels of traffic in advance of a new customer event or deploying a new technology.

Benefits of establishing this best practice: Monitoring and automated management of service quotas and resource constraints can proactively reduce failures. Changes in traffic patterns for a customer's service can cause a disruption or degradation if best practices are not followed. By

monitoring and managing these values across all regions and all accounts, applications can have improved resiliency under adverse or unplanned events.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Service Quotas is an AWS service that helps you manage your quotas for over 250 AWS services from one location. Along with looking up the quota values, you can also request and track quota increases from the Service Quotas console or using the AWS SDK. AWS Trusted Advisor offers a service quotas check that displays your usage and quotas for some aspects of some services. The default service quotas per service are also in the AWS documentation per respective service (for example, see Amazon VPC Quotas).

Some service limits, like rate limits on throttled APIs are set within the Amazon API Gateway itself by configuring a usage plan. Some limits that are set as configuration on their respective services include Provisioned IOPS, Amazon RDS storage allocated, and Amazon EBS volume allocations. Amazon Elastic Compute Cloud has its own service limits dashboard that can help you manage your instance, Amazon Elastic Block Store, and Elastic IP address limits. If you have a use case where service quotas impact your application's performance and they are not adjustable to your needs, then contact Support to see if there are mitigations.

Service quotas can be Region specific or can also be global in nature. Using an AWS service that reaches its quota will not act as expected in normal usage and may cause service disruption or degradation. For example, a service quota limits the number of DL Amazon EC2 instances used in a Region. That limit may be reached during a traffic scaling event using Auto Scaling groups (ASG).

Service quotas for each account should be assessed for usage on a regular basis to determine what the appropriate service limits might be for that account. These service quotas exist as operational guardrails, to prevent accidentally provisioning more resources than you need. They also serve to limit request rates on API operations to protect services from abuse.

Service constraints are different from service quotas. Service constraints represent a particular resource's limits as defined by that resource type. These might be storage capacity (for example, gp2 has a size limit of 1 GB - 16 TB) or disk throughput. It is essential that a resource type's constraint be engineered and constantly assessed for usage that might reach its limit. If a constraint is reached unexpectedly, the account's applications or services may be degraded or disrupted.

If there is a use case where service quotas impact an application's performance and they cannot be adjusted to required needs, contact Support to see if there are mitigations. For more detail on adjusting fixed quotas, see REL01-BP03 Accommodate fixed service quotas and constraints through architecture.

There are a number of AWS services and tools to help monitor and manage Service Quotas. The service and tools should be leveraged to provide automated or manual checks of quota levels.

- AWS Trusted Advisor offers a service quota check that displays your usage and quotas for some aspects of some services. It can aid in identifying services that are near quota.
- AWS Management Console provides methods to display services quota values, manage, request new quotas, monitor status of quota requests, and display history of quotas.
- AWS CLI and CDKs offer programmatic methods to automatically manage and monitor service quota levels and usage.

Implementation steps

For Service Quotas:

- Review AWS Service Quotas.
- To be aware of your existing service quotas, determine the services (like IAM Access Analyzer) that are used. There are approximately 250 AWS services controlled by service quotas. Then, determine the specific service quota name that might be used within each account and Region. There are approximately 3000 service quota names per Region.
- Augment this quota analysis with AWS Config to find all AWS resources used in your AWS accounts.
- Use AWS CloudFormation data to determine your AWS resources used. Look at the resources that were created either in the AWS Management Console or with the `list-stack-resources` AWS CLI command. You can also see resources configured to be deployed in the template itself.
- Determine all the services your workload requires by looking at the deployment code.
- Determine the service quotas that apply. Use the programmatically accessible information from Trusted Advisor and Service Quotas.
- Establish an automated monitoring method (see REL01-BP02 Manage service quotas across accounts and regions and REL01-BP04 Monitor and manage quotas) to alert and inform if services quotas are near or have reached their limit.

- Establish an automated and programmatic method to check if a service quota has been changed in one region but not in other regions in the same account (see REL01-BP02 Manage service quotas across accounts and regions and REL01-BP04 Monitor and manage quotas).
- Automate scanning application logs and metrics to determine if there are any quota or service constraint errors. If these errors are present, send alerts to the monitoring system.
- Establish engineering procedures to calculate the required change in quota (see REL01-BP05 Automate quota management) once it has been identified that larger quotas are required for specific services.
- Create a provisioning and approval workflow to request changes in service quota. This should include an exception workflow in case of request deny or partial approval.
- Create an engineering method to review service quotas prior to provisioning and using new AWS services before rolling out to production or loaded environments. (for example, load testing account).

For service constraints:

- Establish monitoring and metrics methods to alert for resources reading close to their resource constraints. Leverage CloudWatch as appropriate for metrics or log monitoring.
- Establish alert thresholds for each resource that has a constraint that is meaningful to the application or system.
- Create workflow and infrastructure management procedures to change the resource type if the constraint is near utilization. This workflow should include load testing as a best practice to verify that new type is the correct resource type with the new constraints.
- Migrate identified resource to the recommended new resource type, using existing procedures and processes.

Resources

Related best practices:

- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management

- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP04 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

Related tools:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP02 Manage service quotas across accounts and regions

If you are using multiple accounts or Regions, request the appropriate quotas in all environments in which your production workloads run.

Desired outcome: Services and applications should not be affected by service quota exhaustion for configurations that span accounts or Regions or that have resilience designs using zone, Region, or account failover.

Common anti-patterns:

- Allowing resource usage in one isolation Region to grow with no mechanism to maintain capacity in the other ones.
- Manually setting all quotas independently in isolation Regions.

- Not considering the effect of resiliency architectures (like active or passive) in future quota needs during a degradation in the non-primary Region.
- Not evaluating quotas regularly and making necessary changes in every Region and account the workload runs.
- Not leveraging quota request templates to request increases across multiple Regions and accounts.
- Not updating service quotas due to incorrectly thinking that increasing quotas has cost implications like compute reservation requests.

Benefits of establishing this best practice: Verifying that you can handle your current load in secondary regions or accounts if regional services become unavailable. This can help reduce the number of errors or levels of degradations that occur during region loss.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Service quotas are tracked per account. Unless otherwise noted, each quota is AWS Region-specific. In addition to the production environments, also manage quotas in all applicable non-production environments so that testing and development are not hindered. Maintaining a high degree of resiliency requires that service quotas are assessed continually (whether automated or manual).

With more workloads spanning Regions due to the implementation of designs using *Active/Active*, *Active/Passive – Hot*, *Active/Passive-Cold*, and *Active/Passive-Pilot Light* approaches, it is essential to understand all Region and account quota levels. Past traffic patterns are not always a good indicator if the service quota is set correctly.

Equally important, the service quota name limit is not always the same for every Region. In one Region, the value could be five, and in another region the value could be ten. Management of these quotas must span all the same services, accounts, and Regions to provide consistent resilience under load.

Reconcile all the service quota differences across different Regions (Active Region or Passive Region) and create processes to continually reconcile these differences. The testing plans of passive Region failovers are rarely scaled to peak active capacity, meaning that game day or table top exercises can fail to find differences in service quotas between Regions and also then maintain the correct limits.

Service quota drift, the condition where service quota limits for a specific named quota is changed in one Region and not all Regions, is very important to track and assess. Changing the quota in Regions with traffic or potentially could carry traffic should be considered.

- Select relevant accounts and Regions based on your service requirements, latency, regulatory, and disaster recovery (DR) requirements.
- Identify service quotas across all relevant accounts, Regions, and Availability Zones. The limits are scoped to account and Region. These values should be compared for differences.

Implementation steps

- Review Service Quotas values that might have breached beyond the a risk level of usage. AWS Trusted Advisor provides alerts for 80% and 90% threshold breaches.
- Review values for service quotas in any Passive Regions (in an Active/Passive design). Verify that load will successfully run in secondary Regions in the event of a failure in the primary Region.
- Automate assessing if any service quota drift has occurred between Regions in the same account and act accordingly to change the limits.
- If the customer Organizational Units (OU) are structured in the supported manner, service quota templates should be updated to reflect changes in any quotas that should be applied to multiple Regions and accounts.
  - Create a template and associate Regions to the quota change.
  - Review all existing service quota templates for any changes required (Region, limits, and accounts).

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload

- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP04 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas

- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo

Related services:

- Amazon CodeGuru Reviewer
- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP03 Accommodate fixed service quotas and constraints through architecture

Be aware of unchangeable service quotas, service constraints, and physical resource limits. Design architectures for applications and services to prevent these limits from impacting reliability.

Examples include network bandwidth, serverless function invocation payload size, throttle burst rate for of an API gateway, and concurrent user connections to a database.

Desired outcome: The application or service performs as expected under normal and high traffic conditions. They have been designed to work within the limitations for that resource's fixed constraints or service quotas.

Common anti-patterns:

- Choosing a design that uses a resource of a service, unaware that there are design constraints that will cause this design to fail as you scale.
- Performing benchmarking that is unrealistic and will reach service fixed quotas during the testing. For example, running tests at a burst limit but for an extended amount of time.

- Choosing a design that cannot scale or be modified if fixed service quotas are to be exceeded. For example, an SQS payload size of 256KB.
- Observability has not been designed and implemented to monitor and alert on thresholds for service quotas that might be at risk during high traffic events

Benefits of establishing this best practice: Verifying that the application will run under all projected services load levels without disruption or degradation.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Unlike soft service quotas or resources that be replaced with higher capacity units, AWS services' fixed quotas cannot be changed. This means that all these type of AWS services must be evaluated for potential hard capacity limits when used in an application design.

Hard limits are shown in the Service Quotas console. If the columns shows `ADJUSTABLE = No`, the service has a hard limit. Hard limits are also shown in some resources configuration pages. For example, Lambda has specific hard limits that cannot be adjusted.

As an example, when designing a python application to run in a Lambda function, the application should be evaluated to determine if there is any chance of Lambda running longer than 15 minutes. If the code may run more than this service quota limit, alternate technologies or designs must be considered. If this limit is reached after production deployment, the application will suffer degradation and disruption until it can be remediated. Unlike soft quotas, there is no method to change to these limits even under emergency Severity 1 events.

Once the application has been deployed to a testing environment, strategies should be used to find if any hard limits can be reached. Stress testing, load testing, and chaos testing should be part of the introduction test plan.

Implementation steps

- Review the complete list of AWS services that could be used in the application design phase.
- Review the soft quota limits and hard quota limits for all these services. Not all limits are shown in the Service Quotas console. Some services describe these limits in alternate locations.
- As you design your application, review your workload's business and technology drivers, such as business outcomes, use case, dependent systems, availability targets, and disaster recovery

objects. Let your business and technology drivers guide the process to identify the distributed system that is right for your workload.

- Analyze service load across Regions and accounts. Many hard limits are regionally based for services. However, some limits are account based.
- Analyze resilience architectures for resource usage during a zonal failure and Regional failure. In the progression of multi-Region designs using active/active, active/passive – hot, active/passive - cold, and active/passive - pilot light approaches, these failure cases will cause higher usage. This creates a potential use case for hitting hard limits.

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP04 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?

- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Related tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config

- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL01-BP04 Monitor and manage quotas

Evaluate your potential usage and increase your quotas appropriately, allowing for planned growth in usage.

Desired outcome: Active and automated systems that manage and monitor have been deployed. These operations solutions ensure that quota usage thresholds are nearing being reached. These would be proactively remediated by requested quota changes.

Common anti-patterns:

- Not configuring monitoring to check for service quota thresholds
- Not configuring monitoring for hard limits, even though those values cannot be changed.
- Assuming that amount of time required to request and secure a soft quota change is immediate or a short period.
- Configuring alarms for when service quotas are being approached, but having no process on how to respond to an alert.
- Only configuring alarms for services supported by AWS Service Quotas and not monitoring other AWS services.
- Not considering quota management for multiple Region resiliency designs, like active/active, active/passive – hot, active/passive - cold, and active/passive - pilot light approaches.
- Not assessing quota differences between Regions.
- Not assessing the needs in every Region for a specific quota increase request.
- Not leveraging templates for multi-Region quota management.

Benefits of establishing this best practice: Automatic tracking of the AWS Service Quotas and monitoring your usage against those quotas will allow you to see when you are approaching a quota limit. You can also use this monitoring data to help limit any degradations due to quota exhaustion.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

For supported services, you can monitor your quotas by configuring various different services that can assess and then send alerts or alarms. This can aid in monitoring usage and can alert you to approaching quotas. These alarms can be invoked from AWS Config, Lambda functions, Amazon CloudWatch, or from AWS Trusted Advisor. You can also use metric filters on CloudWatch Logs to search and extract patterns in logs to determine if usage is approaching quota thresholds.

Implementation steps

For monitoring:

- Capture current resource consumption (for example, buckets or instances). Use service API operations, such as the Amazon EC2 `DescribeInstances` API, to collect current resource consumption.
- Capture your current quotas that are essential and applicable to the services using:
  - AWS Service Quotas
  - AWS Trusted Advisor
  - AWS documentation
  - AWS service-specific pages
  - AWS Command Line Interface (AWS CLI)
  - AWS Cloud Development Kit (AWS CDK)
- Use AWS Service Quotas, an AWS service that helps you manage your quotas for over 250 AWS services from one location.
- Use Trusted Advisor service limits to monitor your current service limits at various thresholds.
- Use the service quota history (console or AWS CLI) to check on regional increases.
- Compare service quota changes in each Region and each account to create equivalency, if required.

For management:

- Automated: Set up an AWS Config custom rule to scan service quotas across Regions and compare for differences.
- Automated: Set up a scheduled Lambda function to scan service quotas across Regions and compare for differences.

- Manual: Scan services quota through AWS CLI, API, or AWS Console to scan service quotas across Regions and compare for differences. Report the differences.
- If differences in quotas are identified between Regions, request a quota change, if required.
- Review the result of all requests.

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP05 Automate quota management
- REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP04 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide

- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Related tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager

- AWS Marketplace

REL01-BP05 Automate quota management

Service quotas, also referred to as limits in AWS services, are the maximum values for the resources in your AWS account. Each AWS service defines a set of quotas and their default values. To provide your workload access to all the resources it needs, you might need to increase your service quota values.

Growth in workload consumption of AWS resources can threaten workload stability and impact the user experience if quotas are exceeded. Implement tools to alert you when your workload approaches the limits and consider creating quota increase requests automatically.

Desired outcome: Quotas are appropriately configured for the workloads running in each AWS account and Region.

Common anti-patterns:

- You fail to consider and adjust quotas appropriately to meet workload requirements.
- You track quotas and usage using methods that can become outdated, such as with spreadsheets.
- You only update service limits on periodic schedules.
- Your organization lacks operational processes to review existing quotas and request service quota increases when necessary.

Benefits of establishing this best practice:

- Enhanced workload resiliency: You prevent errors caused by exceeding AWS resource quotas.
- Simplified disaster recovery: You can reuse automated quota management mechanisms built in the primary Region during DR setup in another AWS Region.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

View current quotas and track ongoing quota consumption through mechanisms such as AWS Service Quotas console, AWS Command Line Interface (AWS CLI), and AWS SDKs. You can also

integrate your configuration management databases (CMDB) and IT service management (ITSM) systems with the AWS Service Quota APIs.

Generate automated alerts if quota usage reaches your defined thresholds, and define a process for submitting quota increase requests when you receive alerts. If the underlying workload is critical to your business, you can automate quota increase requests, but carefully test the automation to avoid the risk of runaway action such as a growth feedback loop.

Smaller quota increases are often automatically approved. Larger quota requests may need to be manually processed by AWS support and can take additional time to review and process. Allow for additional time to process multiple requests or large increase requests.

Implementation steps

- Implement automated monitoring of service quotas, and issue alerts if your workload's resource utilization growth approaches your quota limits. For example, Quota Monitor for AWS can provide automated monitoring of service quotas. This tool integrates with AWS Organizations and deploys using Cloudformation StackSets so that new accounts are automatically monitored on creation.
- Use features such as Service Quotas request templates or AWS Control Tower to simplify Service Quotas setup for new accounts.
- Build dashboards of your current service quota use across all AWS accounts and regions and reference them as necessary to prevent exceeding your quotas. Trusted Advisor Organizational (TAO) Dashboard, part of the Cloud Intelligence Dashboards, can get you quickly started with such a dashboard.
- Track service limit increase requests. Consolidated Insights from Multiple Accounts(CIMA) can provide an Organization-level view of all your requests.
- Test alert generation and any quota increase request automation by setting lower quota thresholds in non-production accounts. Do not conduct these tests in a production account.

Resources

Related best practices:

- OPS10-BP07 Automate responses to events

Related documents:

- APN Partner: partners that can help with configuration management
- AWS Marketplace: CMDB products that help track limits
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- Quota Monitor Solution on AWS - AWS Solution
- What is Service Quotas?
- What is Service Quotas request templates?

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower

Related tools:

- Quota Monitor for AWS

REL01-BP06 Ensure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover

This article explains how to maintain space between the resource quota and your usage, and how it can benefit your organization. After you finish using a resource, the usage quota may continue to account for that resource. This can result in a failing or inaccessible resource. Prevent resource failure by verifying that your quotas cover the overlap of inaccessible resources and their replacements. Consider cases like network failure, Availability Zone failure, or Region failures when calculating this gap.

Desired outcome: Small or large failures in resources or resource accessibility can be covered within the current service thresholds. Zone failures, network failures, or even Regional failures have been considered in the resource planning.

Common anti-patterns:

- Setting service quotas based on current needs without accounting for failover scenarios.
- Not considering the principals of static stability when calculating the peak quota for a service.

- Not considering the potential of inaccessible resources in calculating total quota needed for each Region.
- Not considering AWS service fault isolation boundaries for some services and their potential abnormal usage patterns.

Benefits of establishing this best practice: When service disruption events impact application availability, use the cloud to implement strategies to recover from these events. An example strategy is creating additional resources to replace inaccessible resources to accommodate failover conditions without exhausting your service limit.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

When evaluating a quota limit, consider failover cases that might occur due to some degradation. Consider the following failover cases.

- A disrupted or inaccessible VPC.
- An inaccessible subnet.
- A degraded Availability Zone that impacts resource accessibility.
- Networking routes or ingress and egress points are blocked or changed.
- A degraded Region that impacts resource accessibility.
- A subset of resources affected by a failure in a Region or an Availability Zone.

The decision to failover is unique for each situation, as the business impact can vary. Address resource capacity planning in the failover location and the resources' quotas before deciding to failover an application or service.

Consider higher than normal peaks of activity when reviewing quotas for each service. These peaks might be related to resources that are inaccessible due to networking or permissions, but are still active. Unterminated active resources count against the service quota limit.

Implementation steps

- Maintain space between your service quota and your maximum usage to accommodate for a failover or loss of accessibility.

- Determine your service quotas. Account for typical deployment patterns, availability requirements, and consumption growth.
- Request quota increases if necessary. Anticipate a wait time for the quota increase request.
- Determine your reliability requirements (also known as your number of nines).
- Understand potential fault scenarios such as loss of a component, an Availability Zone, or a Region.
- Establish your deployment methodology (examples include canary, blue/green, red/black, and rolling).
- Include an appropriate buffer to the current quota limit. An example buffer could be 15%.
- Include calculations for static stability (Zonal and Regional) where appropriate.
- Plan consumption growth and monitor your consumption trends.
- Consider the static stability impact for your most critical workloads. Assess resources conforming to a statically stable system in all Regions and Availability Zones.
- Consider using On-Demand Capacity Reservations to schedule capacity ahead of any failover. This is a useful strategy to implement for critical business schedules to reduce potential risks of obtaining the correct quantity and type of resources during failover.

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP03 Accommodate fixed service quotas and constraints through architecture
- REL01-BP04 Monitor and manage quotas
- REL01-BP05 Automate quota management
- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP04 Test resiliency using chaos engineering

Related documents:

- AWS Well-Architected Framework's Reliability Pillar: Availability
- AWS Service Quotas (formerly referred to as service limits)
- AWS Trusted Advisor Best Practice Checks (see the Service Limits section)
- AWS limit monitor on AWS answers
- Amazon EC2 Service Limits
- What is Service Quotas?
- How to Request Quota Increase
- Service endpoints and quotas
- Service Quotas User Guide
- Quota Monitor for AWS
- AWS Fault Isolation Boundaries
- Availability with redundancy
- AWS for Data
- What is Continuous Integration?
- What is Continuous Delivery?
- APN Partner: partners that can help with configuration management
- Managing the account lifecycle in account-per-tenant SaaS environments on AWS
- Managing and monitoring API throttling in your workloads
- View AWS Trusted Advisor recommendations at scale with AWS Organizations
- Automating Service Limit Increases and Enterprise Support with AWS Control Tower
- Actions, resources, and condition keys for Service Quotas

Related videos:

- AWS Live re:Inforce 2019 - Service Quotas
- View and Manage Quotas for AWS Services Using Service Quotas
- AWS IAM Quotas Demo
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small

Related tools:

- AWS CodeDeploy
- AWS CloudTrail
- Amazon CloudWatch
- Amazon EventBridge
- Amazon DevOps Guru
- AWS Config
- AWS Trusted Advisor
- AWS CDK
- AWS Systems Manager
- AWS Marketplace

REL 2. How do you plan your network topology?

Workloads often exist in multiple environments. These include multiple cloud environments (both publicly accessible and private) and possibly your existing data center infrastructure. Plans must include network considerations such as intra- and intersystem connectivity, public IP address management, private IP address management, and domain name resolution.

Best practices

- REL02-BP01 Use highly available network connectivity for your workload public endpoints
- REL02-BP02 Provision redundant connectivity between private networks in the cloud and on-premises environments
- REL02-BP03 Ensure IP subnet allocation accounts for expansion and availability
- REL02-BP04 Prefer hub-and-spoke topologies over many-to-many mesh
- REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

REL02-BP01 Use highly available network connectivity for your workload public endpoints

Building highly available network connectivity to public endpoints of your workloads can help you reduce downtime due to loss of connectivity and improve the availability and SLA of your workload. To achieve this, use highly available DNS, content delivery networks (CDNs), API gateways, load balancing, or reverse proxies.

Desired outcome: It is critical to plan, build, and operationalize highly available network connectivity for your public endpoints. If your workload becomes unreachable due to a loss in connectivity, even if your workload is running and available, your customers will see your system as down. By combining the highly available and resilient network connectivity for your workload's public endpoints, along with a resilient architecture for your workload itself, you can provide the best possible availability and service level for your customers.

AWS Global Accelerator, Amazon CloudFront, Amazon API Gateway, AWS Lambda Function URLs, AWS AppSync APIs, and Elastic Load Balancing (ELB) all provide highly available public endpoints. Amazon Route 53 provides a highly available DNS service for domain name resolution to verify that your public endpoint addresses can be resolved.

You can also evaluate AWS Marketplace software appliances for load balancing and proxying.

Common anti-patterns:

- Designing a highly available workload without planning out DNS and network connectivity for high availability.
- Using public internet addresses on individual instances or containers and managing the connectivity to them with DNS.
- Using IP addresses instead of domain names for locating services.
- Not testing out scenarios where connectivity to your public endpoints is lost.
- Not analyzing network throughput needs and distribution patterns.
- Not testing and planning for scenarios where internet network connectivity to your public endpoints of your workload might be interrupted.
- Providing content (like web pages, static assets, or media files) to a large geographic area and not using a content delivery network.
- Not planning for distributed denial of service (DDoS) attacks. DDoS attacks risk shutting out legitimate traffic and lowering availability for your users.

Benefits of establishing this best practice: Designing for highly available and resilient network connectivity ensures that your workload is accessible and available to your users.

Level of risk exposed if this best practice is not established: High

Implementation guidance

At the core of building highly available network connectivity to your public endpoints is the routing of the traffic. To verify your traffic is able to reach the endpoints, the DNS must be able to resolve the domain names to their corresponding IP addresses. Use a highly available and scalable Domain Name System (DNS) such as Amazon Route 53 to manage your domain's DNS records. You can also use health checks provided by Amazon Route 53. The health checks verify that your application is reachable, available, and functional, and they can be set up in a way that they mimic your user's behavior, such as requesting a web page or a specific URL. In case of failure, Amazon Route 53 responds to DNS resolution requests and directs the traffic to only healthy endpoints. You can also consider using Geo DNS and Latency Based Routing capabilities offered by Amazon Route 53.

To verify that your workload itself is highly available, use Elastic Load Balancing (ELB). Amazon Route 53 can be used to target traffic to ELB, which distributes the traffic to the target compute instances. You can also use Amazon API Gateway along with AWS Lambda for a serverless solution. Customers can also run workloads in multiple AWS Regions. With multi-site active/active pattern, the workload can serve traffic from multiple Regions. With a multi-site active/passive pattern, the workload serves traffic from the active region while data is replicated to the secondary region and becomes active in the event of a failure in the primary region. Route 53 health checks can then be used to control DNS failover from any endpoint in a primary Region to an endpoint in a secondary Region, verifying that your workload is reachable and available to your users.

Amazon CloudFront provides a simple API for distributing content with low latency and high data transfer rates by serving requests using a network of edge locations around the world. Content delivery networks (CDNs) serve customers by serving content located or cached at a location near to the user. This also improves availability of your application as the load for content is shifted away from your servers over to CloudFront's edge locations. The edge locations and regional edge caches hold cached copies of your content close to your viewers resulting in quick retrieval and increasing reachability and availability of your workload.

For workloads with users spread out geographically, AWS Global Accelerator helps you improve the availability and performance of the applications. AWS Global Accelerator provides Anycast static IP addresses that serve as a fixed entry point to your application hosted in one or more AWS Regions. This allows traffic to ingress onto the AWS global network as close to your users as possible, improving reachability and availability of your workload. AWS Global Accelerator also monitors the health of your application endpoints by using TCP, HTTP, and HTTPS health checks. Any changes in the health or configuration of your endpoints permit redirection of user traffic to healthy endpoints that deliver the best performance and availability to your users. In addition, AWS

Global Accelerator has a fault-isolating design that uses two static IPv4 addresses that are serviced by independent network zones increasing the availability of your applications.

To help protect customers from DDoS attacks, AWS provides AWS Shield Standard. Shield Standard comes automatically turned on and protects from common infrastructure (layer 3 and 4) attacks like SYN/UDP floods and reflection attacks to support high availability of your applications on AWS. For additional protections against more sophisticated and larger attacks (like UDP floods), state exhaustion attacks (like TCP SYN floods), and to help protect your applications running on Amazon Elastic Compute Cloud (Amazon EC2), Elastic Load Balancing (ELB), Amazon CloudFront, AWS Global Accelerator, and Route 53, you can consider using AWS Shield Advanced. For protection against Application layer attacks like HTTP POST or GET floods, use AWS WAF. AWS WAF can use IP addresses, HTTP headers, HTTP body, URI strings, SQL injection, and cross-site scripting conditions to determine if a request should be blocked or allowed.

Implementation steps

1. Set up highly available DNS: Amazon Route 53 is a highly available and scalable domain name system (DNS) web service. Route 53 connects user requests to internet applications running on AWS or on-premises. For more information, see configuring Amazon Route 53 as your DNS service.
2. Setup health checks: When using Route 53, verify that only healthy targets are resolvable. Start by creating Route 53 health checks and configuring DNS failover. The following aspects are important to consider when setting up health checks:
   a. How Amazon Route 53 determines whether a health check is healthy
   b. Creating, updating, and deleting health checks
   c. Monitoring health check status and getting notifications
   d. Best practices for Amazon Route 53 DNS
3. Connect your DNS service to your endpoints.
   a. When using Elastic Load Balancing as a target for your traffic, create an alias record using Amazon Route 53 that points to your load balancer's regional endpoint. During the creation of the alias record, set the Evaluate target health option to Yes.
   b. For serverless workloads or private APIs when API Gateway is used, use Route 53 to direct traffic to API Gateway.
4. Decide on a content delivery network.
   a. For delivering content using edge locations closer to the user, start by understanding how CloudFront delivers content.

   b. Get started with a simple CloudFront distribution. CloudFront then knows where you want the content to be delivered from, and the details about how to track and manage content delivery. The following aspects are important to understand and consider when setting up CloudFront distribution:
      i. How caching works with CloudFront edge locations
      ii. Increasing the proportion of requests that are served directly from the CloudFront caches (cache hit ratio)
      iii. Using Amazon CloudFront Origin Shield
      iv. Optimizing high availability with CloudFront origin failover
5. Set up application layer protection: AWS WAF helps you protect against common web exploits and bots that can affect availability, compromise security, or consume excessive resources. To get a deeper understanding, review how AWS WAF works and when you are ready to implement protections from application layer HTTP POST AND GET floods, review Getting started with AWS WAF. You can also use AWS WAF with CloudFront see the documentation on how AWS WAF works with Amazon CloudFront features.
6. Set up additional DDoS protection: By default, all AWS customers receive protection from common, most frequently occurring network and transport layer DDoS attacks that target your web site or application with AWS Shield Standard at no additional charge. For additional protection of internet-facing applications running on Amazon EC2, Elastic Load Balancing, Amazon CloudFront, AWS Global Accelerator, and Amazon Route 53 you can consider AWS Shield Advanced and review examples of DDoS resilient architectures. To protect your workload and your public endpoints from DDoS attacks review Getting started with AWS Shield Advanced.

Resources

Related best practices:

- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP04 Rely on the data plane and not the control plane during recovery
- REL11-BP06 Send notifications when events impact availability

Related documents:

- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure

- What Is AWS Global Accelerator?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- What is Elastic Load Balancing?
- Network Connectivity capability - Establishing Your Cloud Foundations
- What is Amazon API Gateway?
- What are AWS WAF, AWS Shield, and AWS Firewall Manager?
- What is Amazon Application Recovery Controller?
- Configure custom health checks for DNS failover

Related videos:

- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2020: Global traffic management with Amazon Route 53
- AWS re:Invent 2022 - Operating highly available Multi-AZ applications
- AWS re:Invent 2022 - Dive deep on AWS networking infrastructure
- AWS re:Invent 2022 - Building resilient networks

Related examples:

- Disaster Recovery with Amazon Application Recovery Controller (ARC)
- AWS Global Accelerator Workshop

REL02-BP02 Provision redundant connectivity between private networks in the cloud and on-premises environments

Implement redundancy in your connections between private networks in the cloud and on-premises environments to achieve connectivity resilience. This can be accomplished by deploying two or more links and traffic paths, preserving connectivity in the event of network failures.

Common anti-patterns:

- You depend on just one network connection, which creates a single point of failure.
- You use only one VPN tunnel or multiple tunnels that end in the same Availability Zone.

- You rely on one ISP for VPN connectivity, which can lead to complete failures during ISP outages.
- Not implementing dynamic routing protocols like BGP, which are crucial for rerouting traffic during network disruptions.
- You ignore the bandwidth limitations of VPN tunnels and overestimate their backup capabilities.

Benefits of establishing this best practice: By implementing redundant connectivity between your cloud environment and your corporate or on-premises environment, the dependent services between the two environments can communicate reliably.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When using AWS Direct Connect to connect your on-premises network to AWS, you can achieve maximum network resiliency (SLA of 99.99%) by using separate connections that end on distinct devices in more than one on-premises location and more than one AWS Direct Connect location. This topology offers resilience against device failures, connectivity issues, and complete location outages. Alternatively, you can achieve high resiliency (SLA of 99.9%) by using two individual connections to multiple locations (each on-premises location connected to a single Direct Connect location). This approach protects against connectivity disruptions caused by fiber cuts or device failures and helps mitigate complete location failures. The Direct Connect Resiliency Toolkit can assist in designing your AWS Direct Connect topology.

You can also consider AWS Site-to-Site VPN ending on an AWS Transit Gateway as a cost-effective backup to your primary AWS Direct Connect connection. This setup enables equal-cost multipath (ECMP) routing across multiple VPN tunnels, allowing for throughput of up to 50Gbps, even though each VPN tunnel is capped at 1.25 Gbps. It's important to note, however, that AWS Direct Connect is still the most effective choice for minimizing network disruptions and providing stable connectivity.

When using VPNs over the internet to connect your cloud environment to your on-premises data center, configure two VPN tunnels as part of a single site-to-site VPN connection. Each tunnel should end in a different Availability Zone for high availability and use redundant hardware to prevent on-premises device failure. Additionally, consider multiple internet connections from various internet service providers (ISPs) at your on-premises location to avoid complete VPN connectivity disruption due to a single ISP outage. Selecting ISPs with diverse routing and infrastructure, especially those with separate physical paths to AWS endpoints, provides high connectivity availability.

In addition to physical redundancy with multiple AWS Direct Connect connections and multiple VPN tunnels (or a combination of both), implementing Border Gateway Protocol (BGP) dynamic routing is also crucial. Dynamic BGP provides automatic rerouting of traffic from one path to another based on real-time network conditions and configured policies. This dynamic behavior is especially beneficial in maintaining network availability and service continuity in the event of link or network failures. It quickly selects alternative paths, enhancing the network's resilience and reliability.

Implementation steps

- Acquisition highly-available connectivity between AWS and your on-premises environment.
  - Use multiple AWS Direct Connect connections or VPN tunnels between separately deployed private networks.
  - Use multiple Direct Connect locations for high availability.
  - If using multiple AWS Regions, create redundancy in at least two of them.
- Use AWS Transit Gateway, when possible, to end your VPN connection.
- Evaluate AWS Marketplace appliances to end VPNs or extend your SD-WAN to AWS. If you use AWS Marketplace appliances, deploy redundant instances for high availability in different Availability Zones.
- Provide a redundant connection to your on-premises environment.
  - You may need redundant connections to multiple AWS Regions to achieve your availability needs.
  - Use the Direct Connect Resiliency Toolkit to get started.

Resources

Related documents:

- AWS Direct Connect Resiliency Recommendations
- Using Redundant Site-to-Site VPN Connections to Provide Failover
- Routing policies and BGP communities
- Active/Active and Active/Passive Configurations in AWS Direct Connect
- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options Whitepaper

- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- Using redundant Site-to-Site VPN connections to provide failover
- Using the Direct Connect Resiliency Toolkit to get started
- VPC Endpoints and VPC Endpoint Services (AWS PrivateLink)
- What Is Amazon VPC?
- What is a transit gateway?
- What is AWS Site-to-Site VPN?
- Working with Direct Connect gateways

Related videos:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs

REL02-BP03 Ensure IP subnet allocation accounts for expansion and availability

Amazon VPC IP address ranges must be large enough to accommodate workload requirements, including factoring in future expansion and allocation of IP addresses to subnets across Availability Zones. This includes load balancers, EC2 instances, and container-based applications.

When you plan your network topology, the first step is to define the IP address space itself. Private IP address ranges (following RFC 1918 guidelines) should be allocated for each VPC. Accommodate the following requirements as part of this process:

- Allow IP address space for more than one VPC per Region.
- Within a VPC, allow space for multiple subnets so that you can cover multiple Availability Zones.
- Consider leaving unused CIDR block space within a VPC for future expansion.
- Ensure that there is IP address space to meet the needs of any transient fleets of Amazon EC2 instances that you might use, such as Spot Fleets for machine learning, Amazon EMR clusters, or Amazon Redshift clusters. Similar consideration should be given to Kubernetes clusters, such as Amazon Elastic Kubernetes Service (Amazon EKS), as each Kubernetes pod is assigned a routable address from the VPC CIDR block by default.
- Note that the first four IP addresses and the last IP address in each subnet CIDR block are reserved and not available for your use.

- Note that the initial VPC CIDR block allocated to your VPC cannot be changed or deleted, but you can add additional non-overlapping CIDR blocks to the VPC. Subnet IPv4 CIDRs cannot be changed, however IPv6 CIDRs can.
- The largest possible VPC CIDR block is a /16, and the smallest is a /28.
- Consider other connected networks (VPC, on-premises, or other cloud providers) and ensure non-overlapping IP address space. For more information, see REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected.

Desired outcome: A scalable IP subnet can help you accomodate for future growth and avoid unnecessary waste.

Common anti-patterns:

- Failing to consider future growth, resulting in CIDR blocks that are too small and requiring reconfiguration, potentially causing downtime.
- Incorrectly estimating how many IP addresses an elastic load balancer can use.
- Deploying many high traffic load balancers into the same subnets
- Using automated scaling mechanisms whilst failing to monitor IP address consumption.
- Defining excessively large CIDR ranges well beyond future growth expectations, which can lead to difficulty peering with other networks with overlapping address ranges.

Benefits of establishing this best practice: This ensures that you can accommodate the growth of your workloads and continue to provide availability as you scale up.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Plan your network to accommodate for growth, regulatory compliance, and integration with others. Growth can be underestimated, regulatory compliance can change, and acquisitions or private network connections can be difficult to implement without proper planning.

- Select relevant AWS accounts and Regions based on your service requirements, latency, regulatory, and disaster recovery (DR) requirements.
- Identify your needs for regional VPC deployments.
- Identify the size of the VPCs.

- Determine if you are going to deploy multi-VPC connectivity.
  - What Is a Transit Gateway?
  - Single Region Multi-VPC Connectivity
- Determine if you need segregated networking for regulatory requirements.
- Make VPCs with appropriately-sized CIDR blocks to accommodate your current and future needs.
  - If you have unknown growth projections, you may wish to err on the side of larger CIDR blocks to reduce the potential for future reconfiguration
- Consider using IPv6 addressing for subnets as part of a dual-stack VPC. IPv6 is well suited to being used in private subnets containing fleets of ephemeral instances or containers that would otherwise require large numbers of IPv4 addresses.

Resources

Related Well-Architected best practices:

- REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

Related documents:

- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- Single Region Multi-VPC Connectivity
- What Is Amazon VPC?
- IPv6 on AWS
- IPv6 on reference architectures
- Amazon Elastic Kubernetes Service launches IPv6 support
- Recommendations for your VPC - Classic Load Balancers
- Availability Zone subnets - Application Load Balancers
- Availability Zones - Network Load Balancers

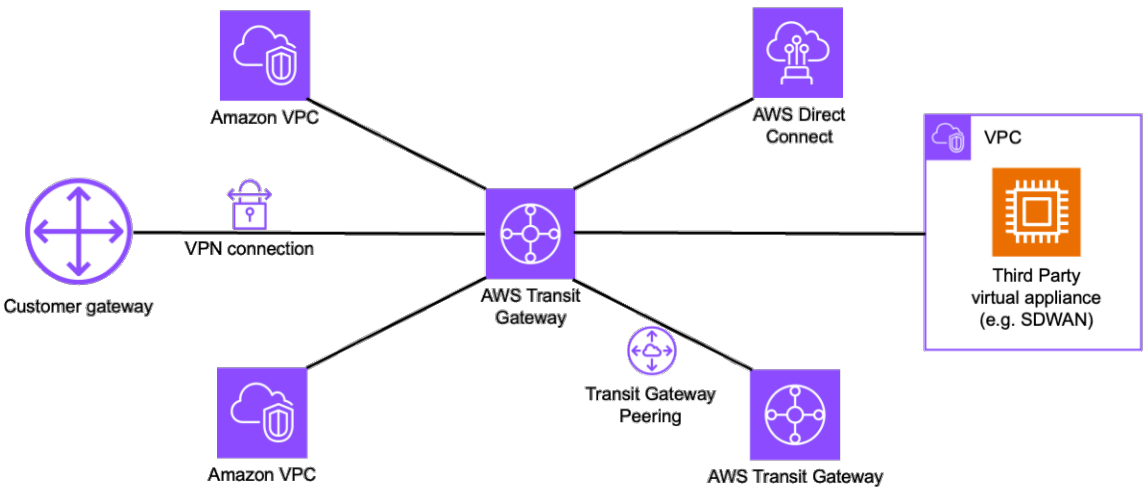
Related videos:

- AWS re:Invent 2018: Advanced VPC Design and New Capabilities for Amazon VPC (NET303)
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs (NET406-R1)
- AWS re:Invent 2023: AWS Ready for what's next? Designing networks for growth and flexibility (NET310)

REL02-BP04 Prefer hub-and-spoke topologies over many-to-many mesh

When connecting multiple private networks, such as Virtual Private Clouds (VPCs) and on-premises networks, opt for a hub-and-spoke topology over a meshed one. Unlike meshed topologies, where each network connects directly to the others and increases the complexity and management overhead, the hub-and-spoke architecture centralizes connections through a single hub. This centralization simplifies the network structure and enhances its operability, scalability, and control.

AWS Transit Gateway is a managed, scalable, and highly-available service designed for construction of hub-and-spoke networks on AWS. It serves as the central hub of your network that provides network segmentation, centralized routing, and the simplified connection to both cloud and on-premises environments. The following figure illustrates how you can use AWS Transit Gateway to build your hub-and-spoke topology.



Desired outcome: You have connected your Virtual Private Clouds (VPCs) and on-premises networks through a central hub. You configure your peering connections through the hub, which acts as a highly scalable cloud router. Routing is simplified because you do not have to work with complex peering relationships. Traffic between networks is encrypted, and you have the ability to isolate networks.

Common anti-patterns:

- You build complex network peering rules.
- You provide routes between networks that should not communicate with one another (for example, separate workloads that have no interdependencies).
- There is ineffective governance of the hub instance.

Benefits of establishing this best practice: As the number of connected networks increases, management and expansion of meshed connectivity becomes increasingly challenging. A mesh architecture introduces additional challenges, such as additional infrastructure components, configuration requirements, and deployment considerations. The mesh also introduces additional overhead to manage and monitor the data plane and control plane components. You must think about how to provide high availability of the mesh architecture, how to monitor the mesh health and performance, and how to handle upgrades of the mesh components.

A hub-and-spoke model, on the other hand, establishes centralized traffic routing across multiple networks. It provides a simpler approach to management and monitoring of the data plane and control plane components.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Create a Network Services account if one does not exist. Place the hub in the organization's Network Services account. This approach allows the hub to be centrally managed by network engineers.

The hub of the hub-and-spoke model acts as a virtual router for traffic flowing between your Virtual Private Clouds (VPCs) and on-premises networks. This approach reduces network complexity and makes it easier to troubleshoot networking issues.

Consider your network design, including the VPCs, AWS Direct Connect, and Site-to-Site VPN connections you want to interconnect.

Consider using a separate subnet for each transit gateway VPC attachment. For each subnet, use a small CIDR (for example /28) so that you have more address space for compute resources. Additionally, create one network ACL, and associate it with all of the subnets that are associated with the hub. Keep the network ACL open in both the inbound and outbound directions.

Design and implement your routing tables such that routes are provided only between networks that should communicate. Omit routes between networks that should not communicate with one another (for example, between separate workloads that have no inter-dependencies).

Implementation steps

1. Plan your network. Determine which networks you want to connect, and verify that they don't share overlapping CIDR ranges.
2. Create an AWS Transit Gateway and attach your VPCs.
3. If needed, create VPN connections or Direct Connect gateways, and associate them with the Transit Gateway.
4. Define how traffic is routed between the connected VPCs and other connections through configuration of your Transit Gateway route tables.
5. Use Amazon CloudWatch to monitor and adjust configurations as necessary for performance and cost optimization.

Resources

Related best practices:

- REL02-BP03 Ensure IP subnet allocation accounts for expansion and availability
- REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

Related documents:

- What Is a Transit Gateway?
- Transit gateway design best practices
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- Building a global network using AWS Transit Gateway Inter-Region peering
- Amazon Virtual Private Cloud Connectivity Options

- APN Partner: partners that can help plan your networking
- AWS Marketplace for Network Infrastructure

Related videos:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities

Related workshops:

- AWS Transit Gateway Workshop

REL02-BP05 Enforce non-overlapping private IP address ranges in all private address spaces where they are connected

The IP address ranges of each of your VPCs must not overlap when peered, connected via Transit Gateway, or connected over VPN. Avoid IP address conflicts between a VPC and on-premises environments or with other cloud providers that you use. You must also have a way to allocate private IP address ranges when needed. An IP address management (IPAM) system can help with automating this.

Desired outcome:

- No IP address range conflicts between VPCs, on-premises environments, or other cloud providers.
- Proper IP address management allows for easier scaling of network infrastructure to accommodate growth and changes in network requirements.

Common anti-patterns:

- Using the same IP range in your VPC as you have on premises, in your corporate network, or other cloud providers
- Not tracking IP ranges of VPCs used to deploy your workloads.
- Relying on manual IP address management processes, such as spreadsheets.
- Over- or under-sizing CIDR blocks, which results in IP address waste or insufficient address space for your workload.

Benefits of establishing this best practice: Active planning of your network will ensure that you do not have multiple occurrences of the same IP address in interconnected networks. This prevents routing problems from occurring in parts of the workload that are using the different applications.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Make use of an IPAM, such as the Amazon VPC IP Address Manager, to monitor and manage your CIDR use. Several IPAMs are also available from the AWS Marketplace. Evaluate your potential usage on AWS, add CIDR ranges to existing VPCs, and create VPCs to allow planned growth in usage.

Implementation steps

- Capture current CIDR consumption (for example, VPCs and subnets).
  - Use service API operations to collect current CIDR consumption.
  - Use the Amazon VPC IP Address Manager to discover resources.
- Capture your current subnet usage.
  - Use service API operations to collect subnets per VPC in each Region.
  - Use the Amazon VPC IP Address Manager to discover resources.
- Record the current usage.
- Determine if you created any overlapping IP ranges.
- Calculate the spare capacity.
- Identify overlapping IP ranges. You can either migrate to a new range of addresses or consider using techniques like private NAT Gateway or AWS PrivateLink if you need to connect the overlapping ranges.

Resources

Related best practices:

- Protecting networks

Related documents:

- APN Partner: partners that can help plan your networking

- AWS Marketplace for Network Infrastructure
- Amazon Virtual Private Cloud Connectivity Options Whitepaper
- Multiple data center HA network connectivity
- Connecting Networks with Overlapping IP Ranges
- What Is Amazon VPC?
- What is IPAM?

Related videos:

- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2019: AWS Transit Gateway reference architectures for many VPCs
- AWS re:Invent 2023 - Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2021 - {New Launch} Manage your IP addresses at scale on AWS

Workload architecture

Questions

- REL 3. How do you design your workload service architecture?
- REL 4. How do you design interactions in a distributed system to prevent failures?
- REL 5. How do you design interactions in a distributed system to mitigate or withstand failures?

REL 3. How do you design your workload service architecture?

Build highly scalable and reliable workloads using a service-oriented architecture (SOA) or a microservices architecture. Service-oriented architecture (SOA) is the practice of making software components reusable via service interfaces. Microservices architecture goes further to make components smaller and simpler.

Best practices

- REL03-BP01 Choose how to segment your workload
- REL03-BP02 Build services focused on specific business domains and functionality
- REL03-BP03 Provide service contracts per API

REL03-BP01 Choose how to segment your workload

Workload segmentation is important when determining the resilience requirements of your application. Monolithic architecture should be avoided whenever possible. Instead, carefully consider which application components can be broken out into microservices. Depending on your application requirements, this may end up being a combination of a service-oriented architecture (SOA) with microservices where possible. Workloads that are capable of statelessness are more capable of being deployed as microservices.

Desired outcome: Workloads should be supportable, scalable, and as loosely coupled as possible.

When making choices about how to segment your workload, balance the benefits against the complexities. What is right for a new product racing to first launch is different than what a workload built to scale from the start needs. When refactoring an existing monolith, you will need to consider how well the application will support a decomposition towards statelessness. Breaking services into smaller pieces allows small, well-defined teams to develop and manage them. However, smaller services can introduce complexities which include possible increased latency, more complex debugging, and increased operational burden.

Common anti-patterns:

- The microservice *Death Star* is a situation in which the atomic components become so highly interdependent that a failure of one results in a much larger failure, making the components as rigid and fragile as a monolith.

Benefits of establishing this practice:

- More specific segments lead to greater agility, organizational flexibility, and scalability.
- Reduced impact of service interruptions.
- Application components may have different availability requirements, which can be supported by a more atomic segmentation.
- Well-defined responsibilities for teams supporting the workload.

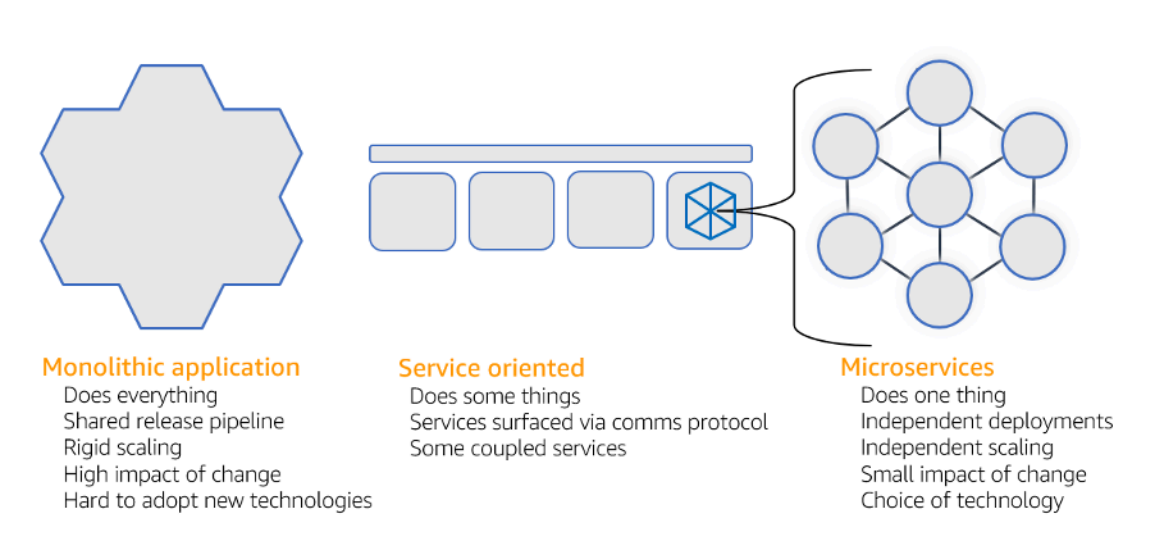
Level of risk exposed if this best practice is not established: High

Implementation guidance

Choose your architecture type based on how you will segment your workload. Choose an SOA or microservices architecture (or in some rare cases, a monolithic architecture). Even if you choose

to start with a monolith architecture, you must ensure that it's modular and can ultimately evolve to SOA or microservices as your product scales with user adoption. SOA and microservices offer respectively smaller segmentation, which is preferred as a modern scalable and reliable architecture, but there are trade-offs to consider, especially when deploying a microservice architecture.

One primary trade-off is that you now have a distributed compute architecture that can make it harder to achieve user latency requirements and there is additional complexity in the debugging and tracing of user interactions. You can use AWS X-Ray to assist you in solving this problem. Another effect to consider is increased operational complexity as you increase the number of applications that you are managing, which requires the deployment of multiple independency components.



Monolithic, service-oriented, and microservices architectures

Implementation steps

- Determine the appropriate architecture to refactor or build your application. SOA and microservices offer respectively smaller segmentation, which is preferred as a modern scalable and reliable architecture. SOA can be a good compromise for achieving smaller segmentation while avoiding some of the complexities of microservices. For more details, see Microservice Trade-Offs.
- If your workload is amenable to it, and your organization can support it, you should use a microservices architecture to achieve the best agility and reliability. For more details, see Implementing Microservices on AWS.

- Consider following the *Strangler Fig* pattern to refactor a monolith into smaller components. This involves gradually replacing specific application components with new applications and services. AWS Migration Hub Refactor Spaces acts as the starting point for incremental refactoring. For more details, see Seamlessly migrate on-premises legacy workloads using a strangler pattern.
- Implementing microservices may require a service discovery mechanism to allow these distributed services to communicate with each other. AWS App Mesh can be used with service-oriented architectures to provide reliable discovery and access of services. AWS Cloud Map can also be used for dynamic, DNS-based service discovery.
- If you're migrating from a monolith to SOA, Amazon MQ can help bridge the gap as a service bus when redesigning legacy applications in the cloud.
- For existing monoliths with a single, shared database, choose how to reorganize the data into smaller segments. This could be by business unit, access pattern, or data structure. At this point in the refactoring process, you should choose to move forward with a relational or non-relational (NoSQL) type of database. For more details, see From SQL to NoSQL.

Level of effort for the implementation plan: High

Resources

Related best practices:

- REL03-BP02 Build services focused on specific business domains and functionality

Related documents:

- Amazon API Gateway: Configuring a REST API Using OpenAPI
- What is Service-Oriented Architecture?
- Bounded Context (a central pattern in Domain-Driven Design)
- Implementing Microservices on AWS
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- Microservices on AWS
- What is AWS App Mesh?

Related examples:

- Iterative App Modernization Workshop

Related videos:

- Delivering Excellence with Microservices on AWS

REL03-BP02 Build services focused on specific business domains and functionality

Service-oriented architectures (SOA) define services with well-delineated functions defined by business needs. Microservices use domain models and bounded context to draw service boundaries along business context boundaries. Focusing on business domains and functionality helps teams define independent reliability requirements for their services. Bounded contexts isolate and encapsulate business logic, allowing teams to better reason about how to handle failures.

Desired outcome: Engineers and business stakeholders jointly define bounded contexts and use them to design systems as services that fulfill specific business functions. These teams use established practices like event storming to define requirements. New applications are designed as services well-defined boundaries and loosely coupling. Existing monoliths are decomposed into bounded contexts and system designs move towards SOA or microservice architectures. When monoliths are refactored, established approaches like bubble contexts and monolith decomposition patterns are applied.

Domain-oriented services are executed as one or more processes that don't share state. They independently respond to fluctuations in demand and handle fault scenarios in light of domain specific requirements.

Common anti-patterns:

- Teams are formed around specific technical domains like UI and UX, middleware, or database instead of specific business domains.
- Applications span domain responsibilities. Services that span bounded contexts can be more difficult to maintain, require larger testing efforts, and require multiple domain teams to participate in software updates.
- Domain dependencies, like domain entity libraries, are shared across services such that changes for one service domain require changes to other service domains

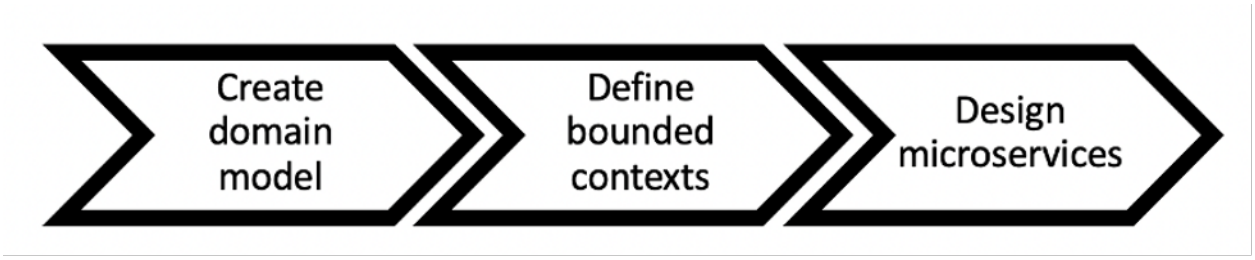
- Service contracts and business logic don't express entities in a common and consistent domain language, resulting in translation layers that complicate systems and increase debugging efforts.

Benefits of establishing this best practice: Applications are designed as independent services bounded by business domains and use a common business language. Services are independently testable and deployable. Services meet domain specific resiliency requirements for the domain implemented.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Domain-driven design (DDD) is the foundational approach of designing and building software around business domains. It's helpful to work with an existing framework when building services focused on business domains. When working with existing monolithic applications, you can take advantage of decomposition patterns that provide established techniques to modernize applications into services.



Domain-driven design

Implementation steps

- Teams can hold event storming workshops to quickly identify events, commands, aggregates and domains in a lightweight sticky note format.
- Once domain entities and functions have been formed in a domain context, you can divide your domain into services using bounded context, where entities that share similar features and attributes are grouped together. With the model divided into contexts, a template for how to boundary microservices emerges.
  - For example, the Amazon.com website entities might include package, delivery, schedule, price, discount, and currency.
  - Package, delivery, and schedule are grouped into the shipping context, while price, discount, and currency are grouped into the pricing context.

- Decomposing monoliths into microservices outlines patterns for refactoring microservices. Using patterns for decomposition by business capability, subdomain, or transaction aligns well with domain-driven approaches.
- Tactical techniques such as the bubble context allow you to introduce DDD in existing or legacy applications without up-front rewrites and full commitments to DDD. In a bubble context approach, a small bounded context is established using a service mapping and coordination, or anti-corruption layer, which protects the newly defined domain model from external influences.

After teams have performed domain analysis and defined entities and service contracts, they can take advantage of AWS services to implement their domain-driven design as cloud-based services.

- Start your development by defining tests that exercise business rules of your domain. Test-driven development (TDD) and behavior-driven development (BDD) help teams keep services focused on solving business problems.
- Select the AWS services that best meet your business domain requirements and microservice architecture:
  - AWS Serverless allows your team focus on specific domain logic instead of managing servers and infrastructure.
  - Containers at AWS simplify the management of your infrastructure, so you can focus on your domain requirements.
  - Purpose built databases help you match your domain requirements to the best fit database type.
- Building hexagonal architectures on AWS outlines a framework to build business logic into services working backwards from a business domain to fulfill functional requirements and then attach integration adapters. Patterns that separate interface details from business logic with AWS services help teams focus on domain functionality and improve software quality.

Resources

Related best practices:

- REL03-BP01 Choose how to segment your workload
- REL03-BP03 Provide service contracts per API

Related documents:

- AWS Microservices
- Implementing Microservices on AWS
- How to break a Monolith into Microservices
- Getting Started with DDD when Surrounded by Legacy Systems
- Domain-Driven Design: Tackling Complexity in the Heart of Software
- Building hexagonal architectures on AWS
- Decomposing monoliths into microservices
- Event Storming
- Messages Between Bounded Contexts
- Microservices
- Test-driven development
- Behavior-driven development

Related examples:

- Designing Cloud Native Microservices on AWS (from DDD/EventStormingWorkshop)

Related tools:

- AWS Cloud Databases
- Serverless on AWS
- Containers at AWS

REL03-BP03 Provide service contracts per API

Service contracts are documented agreements between API producers and consumers defined in a machine-readable API definition. A contract versioning strategy allows consumers to continue using the existing API and migrate their applications to a newer API when they are ready. Producer deployment can happen any time as long as the contract is followed. Service teams can use the technology stack of their choice to satisfy the API contract.

Desired outcome: Applications built with service-oriented or microservice architectures are able to operate independently while having integrated runtime dependency. Changes deployed to an API

consumer or producer do not interrupt the stability of the overall system when both sides follow a common API contract. Components that communicate over service APIs can perform independent functional releases, upgrades to runtime dependencies, or fail over to a disaster recovery (DR) site with little or no impact to each other. In addition, discrete services are able to independently scale absorbing resource demand without requiring other services to scale in unison.

Common anti-patterns:

- Creating service APIs without strongly typed schemas. This results in APIs that cannot be used to generate API bindings and payloads that can't be programmatically validated.
- Not adopting a versioning strategy, which forces API consumers to update and release or fail when service contracts evolve.
- Error messages that leak details of the underlying service implementation rather than describe integration failures in the domain context and language.
- Not using API contracts to develop test cases and mock API implementations to allow for independent testing of service components.

Benefits of establishing this best practice: Distributed systems composed of components that communicate over API service contracts can improve reliability. Developers can catch potential issues early in the development process with type checking during compilation to verify that requests and responses follow the API contract and required fields are present. API contracts provide a clear self-documenting interface for APIs and provider better interoperability between different systems and programming languages.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Once you have identified business domains and determined your workload segmentation, you can develop your service APIs. First, define machine-readable service contracts for APIs, and then implement an API versioning strategy. When you are ready to integrate services over common protocols like REST, GraphQL, or asynchronous events, you can incorporate AWS services into your architecture to integrate your components with strongly-typed API contracts.

AWS services for service API contrats

Incorporate AWS services including Amazon API Gateway, AWS AppSync, and Amazon EventBridge into your architecture to use API service contracts in your application. Amazon API Gateway helps

you integrate with directly native AWS services and other web services. API Gateway supports the OpenAPI specification and versioning. AWS AppSync is a managed GraphQL endpoint you configure by defining a GraphQL schema to define a service interface for queries, mutations and subscriptions. Amazon EventBridge uses event schemas to define events and generate code bindings for your events.

Implementation steps

- First, define a contract for your API. A contract will express the capabilities of an API as well as define strongly typed data objects and fields for the API input and output.
- When you configure APIs in API Gateway, you can import and export OpenAPI Specifications for your endpoints.
  - Importing an OpenAPI definition simplifies the creation of your API and can be integrated with AWS infrastructure as code tools like the AWS Serverless Application Model and AWS Cloud Development Kit (AWS CDK).
  - Exporting an API definition simplifies integrating with API testing tools and provides services consumer an integration specification.
- You can define and manage GraphQL APIs with AWS AppSync by defining a GraphQL schema file to generate your contract interface and simplify interaction with complex REST models, multiple database tables or legacy services.
- AWS Amplify projects that are integrated with AWS AppSync generate strongly typed JavaScript query files for use in your application as well as an AWS AppSync GraphQL client library for Amazon DynamoDB tables.
- When you consume service events from Amazon EventBridge, events adhere to schemas that already exist in the schema registry or that you define with the OpenAPI Spec. With a schema defined in the registry, you can also generate client bindings from the schema contract to integrate your code with events.
- Extending or version your API. Extending an API is a simpler option when adding fields that can be configured with optional fields or default values for required fields.
  - JSON based contracts for protocols like REST and GraphQL can be a good fit for contract extension.
  - XML based contracts for protocols like SOAP should be tested with service consumers to determine the feasibility of contract extension.
- When versioning an API, consider implementing proxy versioning where a facade is used to support versions so that logic can be maintained in a single codebase.

- With API Gateway you can use request and response mappings to simplify absorbing contract changes by establishing a facade to provide default values for new fields or to strip removed fields from a request or response. With this approach the underlying service can maintain a single codebase.

Resources

Related best practices:

- REL03-BP01 Choose how to segment your workload
- REL03-BP02 Build services focused on specific business domains and functionality
- REL04-BP02 Implement loosely coupled dependencies
- REL05-BP03 Control and limit retry calls
- REL05-BP05 Set client timeouts

Related documents:

- What Is An API (Application Programming Interface)?
- Implementing Microservices on AWS
- Microservice Trade-Offs
- Microservices - a definition of this new architectural term
- Microservices on AWS
- Working with API Gateway extensions to OpenAPI
- OpenAPI-Specification
- GraphQL: Schemas and Types
- Amazon EventBridge code bindings

Related examples:

- Amazon API Gateway: Configuring a REST API Using OpenAPI
- Amazon API Gateway to Amazon DynamoDB CRUD application using OpenAPI
- Modern application integration patterns in a serverless age: API Gateway Service Integration
- Implementing header-based API Gateway versioning with Amazon CloudFront
- AWS AppSync: Building a client application

Related videos:

- Using OpenAPI in AWS SAM to manage API Gateway

Related tools:

- Amazon API Gateway
- AWS AppSync
- Amazon EventBridge

REL 4. How do you design interactions in a distributed system to prevent failures?

Distributed systems rely on communications networks to interconnect components, such as servers or services. Your workload must operate reliably despite data loss or latency in these networks. Components of the distributed system must operate in a way that does not negatively impact other components or the workload. These best practices prevent failures and improve mean time between failures (MTBF).

Best practices

- REL04-BP01 Identify the kind of distributed systems you depend on
- REL04-BP02 Implement loosely coupled dependencies
- REL04-BP03 Do constant work
- REL04-BP04 Make mutating operations idempotent

REL04-BP01 Identify the kind of distributed systems you depend on

Distributed systems can be synchronous, asynchronous, or batch. Synchronous systems must process requests as quickly as possible and communicate with each other by making synchronous request and response calls using HTTP/S, REST, or remote procedure call (RPC) protocols. Asynchronous systems communicate with each other by exchanging data asynchronously through an intermediary service without coupling individual systems. Batch systems receive a large volume of input data, run automated data processes without human intervention, and generate output data.

Desired outcome: Design a workload that effectively interacts with synchronous, asynchronous, and batch dependencies.

Common anti-patterns:

- Workload waits indefinitely for a response from its dependencies, which could lead to workload clients timing out, not knowing if their request has been received.
- Workload uses a chain of dependent systems that call each other synchronously. This requires each system to be available and to successfully process a request before the whole chain can succeed, leading to potentially brittle behavior and overall availability.
- Workload communicates with its dependencies asynchronously and rely on the concept of exactly-once guaranteed delivery of messages, when often it is still possible to receive duplicate messages.
- Workload does not use proper batch scheduling tools and allows concurrent execution of the same batch job.

Benefits of establishing this best practice: It is common for a given workload to implement one or more style of communication between synchronous, asynchronous, and batch. This best practice helps you identify the different trade-offs associated with each style of communication to make your workload able to tolerate disruptions in any of its dependencies.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The following sections contain both general and specific implementation guidance for each kind of dependency.

General guidance

- Make sure that the performance and reliability service-level objectives (SLOs) that your dependencies offer meet the performance and reliability requirements of your workload.
- Use AWS observability services to monitor response times and error rates to make sure your dependency is providing service at the levels needed by your workload.
- Identify the potential challenges that your workload may face when communicating with its dependencies. Distributed systems come with a wide range of challenges that might increase architectural complexity, operational burden, and cost. Common challenges include latency, network disruptions, data loss, scaling, and data replication lag.
- Implement robust error handling and logging to help you troubleshoot problems when your dependency experiences issues.

Synchronous dependency

In synchronous communications, your workload sends a request to its dependency and blocks the operation waiting for a response. When its dependency receives the request, it tries to handle it as soon as possible and sends a response back to your workload. A significant challenge with synchronous communication is that it causes temporal coupling, which requires your workload and its dependencies to be available at the same time. When your workload needs to communicate synchronously with its dependencies, consider the following guidance:

- Your workload should not rely on multiple synchronous dependencies to perform a single function. This chain of dependencies increases overall brittleness because all dependencies in the pathway need to be available in order for the request to complete successfully.
- When a dependency is unhealthy or unavailable, determine your error handling and retry strategies. Avoid using bimodal behavior. Bimodal behavior is when your workload exhibits different behavior under normal and failure modes. For more details on bimodal behavior, see REL11-BP05 Use static stability to prevent bimodal behavior.
- Keep in mind that failing fast is better than making your workload wait. For instance, the AWS Lambda Developer Guide describes how to handle retries and failures when you invoke Lambda functions.
- Set timeouts when your workload calls its dependency. This technique avoids waiting too long or waiting indefinitely for a response. For helpful discussion of this issue, see Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications.
- Minimize the number of calls made from your workload to its dependency to fulfill a single request. Having chatty calls between them increases coupling and latency.

Asynchronous dependency

To temporally decouple your workload from its dependency, they should communicate asynchronously. Using an asynchronous approach, your workload can continue with any other processing without having to wait for its dependency, or chain of dependencies, to send a response.

When your workload needs to communicate asynchronously with its dependency, consider the following guidance:

- Determine whether to use messaging or event streaming based on your use case and requirements. Messaging allows your workload to communicate with its dependency by sending

and receiving messages through a message broker. Event streaming allows your workload and its dependency to use a streaming service to publish and subscribe to events, delivered as continuous streams of data, that need to be processed as soon as possible.

- Messaging and event streaming handle messages differently so you need to make trade-off decisions based on:
  - **Message priority:** message brokers can process high-priority messages ahead of normal messages. In event streaming, all messages have the same priority.
  - **Message consumption**: message brokers ensure that consumers receive the message. Event streaming consumers must keep track of the last message they have read.
  - **Message ordering**: with messaging, receiving messages in the exact order they are sent is not guaranteed unless you use a first-in-first-out (FIFO) approach. Event streaming always preserves the order in which the data was produced.
  - **Message deletion**: with messaging, the consumer must delete the message after processing it. The event streaming service appends the message to a stream and remains in there until the message's retention period expires. This deletion policy makes event streaming suitable for replaying messages.
- Define how your workload knows when its dependency completes its work. For instance, when your workload invokes a Lambda function asynchronously, Lambda places the event in a queue and returns a success response without additional information. After processing is complete, the Lambda function can send the result to a destination, configurable based on success or failure.
- Build your workload to handle duplicate messages by leveraging idempotency. Idempotency means that the results of your workload do not change even if your workload is generated more than once for the same message. It is important to point out that messaging or streaming services will redeliver a message if a network failure occurs or if an acknowledgement has not been received.
- If your workload does not get a response from its dependency, it needs to resubmit the request. Consider limiting the number of retries to preserve your workload's CPU, memory, and network resources to handle other requests. The AWS Lambda documentation shows how to handle errors for asynchronous invocation.
- Leverage suitable observability, debugging, and tracing tools to manage and operate your workload's asynchronous communication with its dependency. You can use Amazon CloudWatch to monitor messaging and event streaming services. You can also instrument your workload with AWS X-Ray to quickly gain insights for troubleshooting problems.

Batch dependency

Batch systems take input data, initiate a series of jobs to process it, and produce some output data, without manual intervention. Depending on the data size, jobs could run from minutes to, in some cases, several days. When your workload communicates with its batch dependency, consider the following guidance:

- Define the time window when your workload should run the batch job. Your workload can set up a recurrence pattern to invoke a batch system, for example, every hour or at the end of every month.
- Determine the location of the data input and the processed data output. Choose a storage service, such as Amazon Simple Storage Services (Amazon S3), Amazon Elastic File System (Amazon EFS), and Amazon FSx for Lustre, that allows your workload to read and write files at scale.
- If your workload needs to invoke multiple batch jobs, you could leverage AWS Step Functions to simplify the orchestration of batch jobs that run in AWS or on-premises. This sample project demonstrates orchestration of batch jobs using Step Functions, AWS Batch, and Lambda.
- Monitor batch jobs to look for abnormalities, such as a job taking longer than it should to complete. You could use tools like CloudWatch Container Insights to monitor AWS Batch environments and jobs. In this instance, your workload would stop the next job from beginning and inform the relevant staff of the exception.

Resources

Related documents:

- AWS Cloud Operations: Monitoring and Observability
- The Amazon's Builder Library: Challenges with distributed systems
- REL11-BP05 Use static stability to prevent bimodal behavior
- AWS Lambda Developer Guide: Error handling and automatic retries in AWS Lambda
- Tuning AWS Java SDK HTTP request settings for latency-aware Amazon DynamoDB applications
- AWS Messaging
- What is streaming data?
- AWS Lambda Developer Guide: Asynchronous invocation
- Amazon Simple Queue Service FAQ: FIFO queues

- Amazon Kinesis Data Streams Developer Guide: Handling Duplicate Records
- Amazon Simple Queue Service Developer Guide: Available CloudWatch metrics for Amazon SQS
- Amazon Kinesis Data Streams Developer Guide: Monitoring the Amazon Kinesis Data Streams Service with Amazon CloudWatch
- AWS X-Ray Developer Guide: AWS X-Ray concepts
- AWS Samples on GitHub: AWS Step functions Complex Orchestrator App
- AWS Batch User Guide: AWS Batch CloudWatch Container Insights

Related videos:

- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS (COP310)

Related tools:

- Amazon CloudWatch
- Amazon CloudWatch Logs
- AWS X-Ray
- Amazon Simple Storage Services (Amazon S3)
- Amazon Elastic File System (Amazon EFS)
- Amazon FSx for Lustre
- AWS Step Functions
- AWS Batch

REL04-BP02 Implement loosely coupled dependencies

Dependencies such as queuing systems, streaming systems, workflows, and load balancers are loosely coupled. Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility.

Decoupling dependencies, such as queuing systems, streaming systems, and workflows, help minimize the impact of changes or failure on a system. This separation isolates a component's behavior from affecting others that depend on it, improving resilience and agility.

In tightly coupled systems, changes to one component can necessitate changes in other components that rely on it, resulting in degraded performance across all components. *Loose*

coupling breaks this dependency so that dependent components only need to know the versioned and published interface. Implementing loose coupling between dependencies isolates a failure in one from impacting another.

Loose coupling allows you to modify code or add features to a component while minimizing risk to other components that depend on it. It also allows for granular resilience at a component level where you can scale out or even change underlying implementation of the dependency.

To further improve resiliency through loose coupling, make component interactions asynchronous where possible. This model is suitable for any interaction that does not need an immediate response and where an acknowledgment that a request has been registered will suffice. It involves one component that generates events and another that consumes them. The two components do not integrate through direct point-to-point interaction but usually through an intermediate durable storage layer, such as an Amazon SQS queue, a streaming data platform such as Amazon Kinesis, or AWS Step Functions.

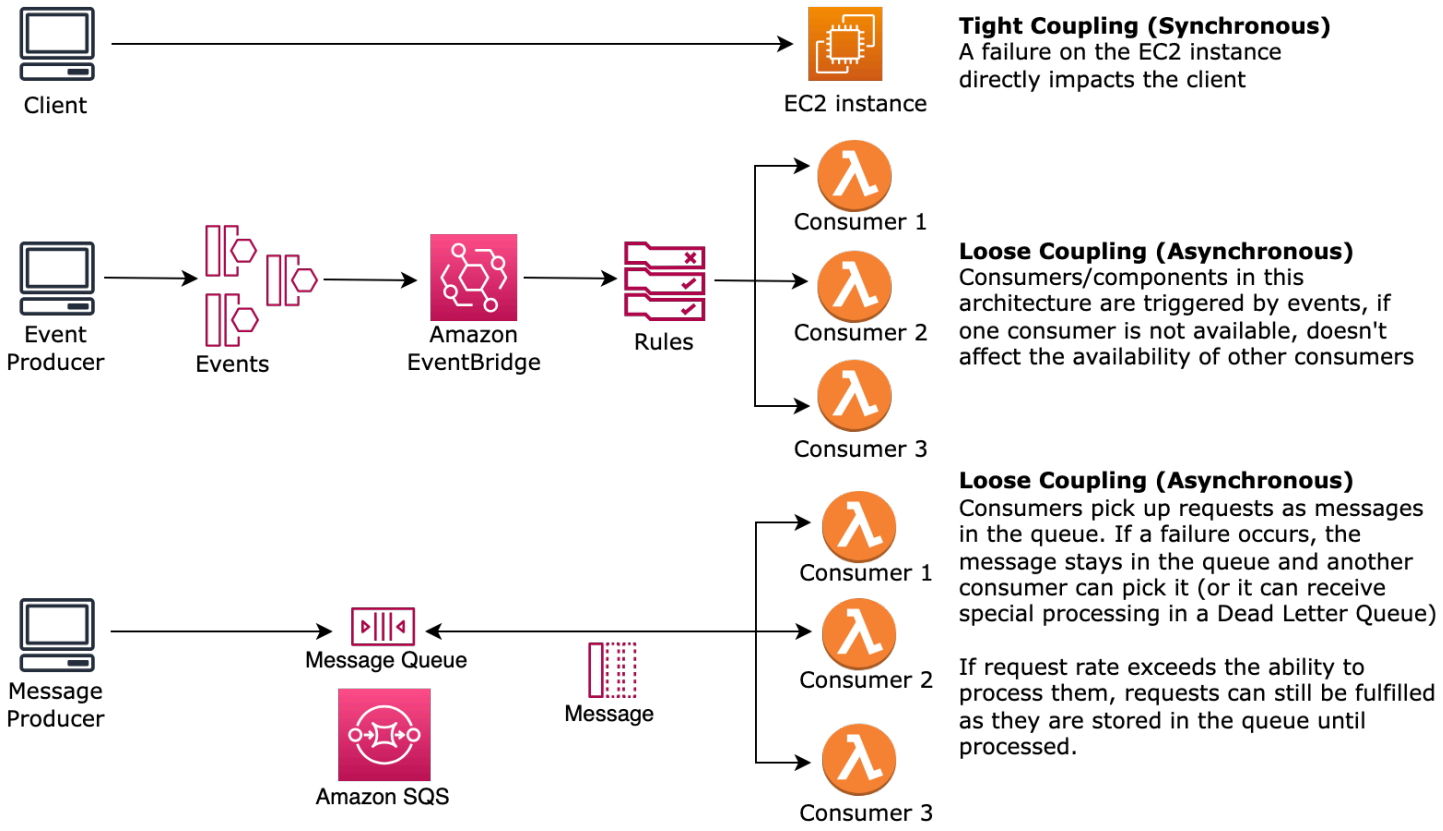


Figure 4: Dependencies such as queuing systems and load balancers are loosely coupled

Amazon SQS queues and AWS Step Functions are just two ways to add an intermediate layer for loose coupling. Event-driven architectures can also be built in the AWS Cloud using Amazon

EventBridge, which can abstract clients (event producers) from the services they rely on (event consumers). Amazon Simple Notification Service (Amazon SNS) is an effective solution when you need high-throughput, push-based, many-to-many messaging. Using Amazon SNS topics, your publisher systems can fan out messages to a large number of subscriber endpoints for parallel processing.

While queues offer several advantages, in most hard real-time systems, requests older than a threshold time (often seconds) should be considered stale (the client has given up and is no longer waiting for a response), and not processed. This way more recent (and likely still valid requests) can be processed instead.

Desired outcome: Implementing loosely coupled dependencies allows you to minimize the surface area for failure to a component level, which helps diagnose and resolve issues. It also simplifies development cycles, allowing teams to implement changes at a modular level without affecting the performance of other components that depend on it. This approach provides the capability to scale out at a component level based on resource needs, as well as utilization of a component contributing to cost-effectiveness.

Common anti-patterns:

- Deploying a monolithic workload.
- Directly invoking APIs between workload tiers with no capability of failover or asynchronous processing of the request.
- Tight coupling using shared data. Loosely coupled systems should avoid sharing data through shared databases or other forms of tightly coupled data storage, which can reintroduce tight coupling and hinder scalability.
- Ignoring back pressure. Your workload should have the ability to slow down or stop incoming data when a component can't process it at the same rate.

Benefits of establishing this best practice: Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility. Failure in one component is isolated from others.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implement loosely coupled dependencies. There are various solutions that allow you to build loosely coupled applications. These include services for implementing fully managed queues,

automated workflows, react to events, and APIs among others which can help isolate behavior of components from other components, and as such increasing resilience and agility.

- **Build event-driven architectures:** Amazon EventBridge helps you build loosely coupled and distributed event-driven architectures.
- **Implement queues in distributed systems:** You can use Amazon Simple Queue Service (Amazon SQS) to integrate and decouple distributed systems.
- **Containerize components as microservices:** Microservices allow teams to build applications composed of small independent components which communicate over well-defined APIs. Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS) can help you get started faster with containers.
- **Manage workflows with Step Functions:** Step Functions help you coordinate multiple AWS services into flexible workflows.
- **Leverage publish-subscribe (pub/sub) messaging architectures:** Amazon Simple Notification Service (Amazon SNS) provides message delivery from publishers to subscribers (also known as producers and consumers).

Implementation steps

- Components in an event-driven architecture are initiated by events. Events are actions that happen in a system, such as a user adding an item to a cart. When an action is successful, an event is generated that actuates the next component of the system.
  - Building Event-driven Applications with Amazon EventBridge
  - AWS re:Invent 2022 - Designing Event-Driven Integrations using Amazon EventBridge
- Distributed messaging systems have three main parts that need to be implemented for a queue based architecture. They include components of the distributed system, the queue that is used for decoupling (distributed on Amazon SQS servers), and the messages in the queue. A typical system has producers which initiate the message into the queue, and the consumer which receives the message from the queue. The queue stores messages across multiple Amazon SQS servers for redundancy.
  - Basic Amazon SQS architecture
  - Send Messages Between Distributed Applications with Amazon Simple Queue Service
- Microservices, when well-utilized, enhance maintainability and boost scalability, as loosely coupled components are managed by independent teams. It also allows for the isolation of behaviors to a single component in case of changes.

  - Implementing Microservices on AWS
  - Let's Architect! Architecting microservices with containers
- With AWS Step Functions you can build distributed applications, automate processes, orchestrate microservices, among other things. The orchestration of multiple components into an automated workflow allows you to decouple dependencies in your application.
  - Create a Serverless Workflow with AWS Step Functions and AWS Lambda
  - Getting Started with AWS Step Functions

Resources

Related documents:

- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: Challenges with distributed systems
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- What Is Amazon EventBridge?
- What Is Amazon Simple Queue Service?
- Break up with your monolith
- Orchestrate Queue-based Microservices with AWS Step Functions and Amazon SQS
- Basic Amazon SQS architecture
- Queue-Based Architecture

Related videos:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)
- AWS re:Invent 2019: Scalable serverless event-driven applications using Amazon SQS and Lambda
- AWS re:Invent 2022 - Designing event-driven integrations using Amazon EventBridge
- AWS re:Invent 2017: Elastic Load Balancing Deep Dive and Best Practices

REL04-BP03 Do constant work

Systems can fail when there are large, rapid changes in load. For example, if your workload is doing a health check that monitors the health of thousands of servers, it should send the same size payload (a full snapshot of the current state) each time. Whether no servers are failing, or all of them, the health check system is doing constant work with no large, rapid changes.

For example, if the health check system is monitoring 100,000 servers, the load on it is nominal under the normally light server failure rate. However, if a major event makes half of those servers unhealthy, then the health check system would be overwhelmed trying to update notification systems and communicate state to its clients. So instead the health check system should send the full snapshot of the current state each time. 100,000 server health states, each represented by a bit, would only be a 12.5-KB payload. Whether no servers are failing, or all of them are, the health check system is doing constant work, and large, rapid changes are not a threat to the system stability. This is actually how Amazon Route 53 handles health checks for endpoints (such as IP addresses) to determine how end users are routed to them.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

- Do constant work so that systems do not fail when there are large, rapid changes in load.
- Implement loosely coupled dependencies. Dependencies such as queuing systems, streaming systems, workflows, and load balancers are loosely coupled. Loose coupling helps isolate behavior of a component from other components that depend on it, increasing resiliency and agility.
  - The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
  - AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
    - For the example of a health check system monitoring 100,000 servers, engineer workloads so that payload sizes remain constant regardless of number of successes or failures.

Resources

Related documents:

- Amazon EC2: Ensuring Idempotency
- The Amazon Builders' Library: Challenges with distributed systems

- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee

Related videos:

- AWS New York Summit 2019: Intro to Event-driven Architectures and Amazon EventBridge (MAD205)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes constant work)
- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019: Moving to event-driven architectures (SVS308)

REL04-BP04 Make mutating operations idempotent

An idempotent service promises that each request is processed exactly once, such that making multiple identical requests has the same effect as making a single request. This makes it easier for a client to implement retries without fear that a request is erroneously processed multiple times. To do this, clients can issue API requests with an idempotency token, which is used whenever the request is repeated. An idempotent service API uses the token to return a response identical to the response that was returned the first time that the request was completed, even if the underlying state of the system has changed.

In a distributed system, it is relatively simple to perform an action at most once (client makes only one request) or at least once (keep requesting until client gets confirmation of success). It is more difficult to guarantee an action is performed *exactly once*, such that making multiple identical requests has the same effect as making a single request. Using idempotency tokens in APIs, services can receive a mutating request one or more times without the need to create duplicate records or side effects.

Desired outcome: You have a consistent, well-documented, and widely adopted approach for ensuring idempotency across all components and services.

Common anti-patterns:

- You apply idempotency indiscriminately, even when not needed.
- You introduce overly complex logic for implementing idempotency.

- You use timestamps as keys for idempotency. This can cause inaccuracies due to clock skew or due to multiple clients that use the same timestamps to apply changes.
- You store entire payloads for idempotency. In this approach, you save complete data payloads for every request and overwrite it at each new request. This can degrade performance and affect scalability.
- You generate keys inconsistently across services. Without consistent keys, services may fail to recognize duplicate requests, which results in unintended results.

Benefits of establishing this best practice:

- Greater scalability: The system can handle retries and duplicate requests without having to perform additional logic or complex state management.
- Enhanced reliability: Idempotency helps services handle multiple identical requests in a consistent manner, which reduces the risk of unintended side effects or duplicate records. This is especially crucial in distributed systems, where network failures and retries are common.
- Improved data consistency: Because the same request produces the same response, idempotency helps maintain data consistency across distributed systems. This is essential to maintain the integrity of transactions and operations.
- Error handling: Idempotency tokens make error handling more straightforward. If a client does not receive a response due to an issue, it can safely resend the request with the same idempotency token.
- Operational transparency: Idempotency allows for better monitoring and logging. Services can log requests with their idempotency tokens, which makes it easier to trace and debug issues.
- Simplified API contract: It can simplify the contract between the client and server side systems and reduce the fear of erroneous data processing.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

In a distributed system, performing an action at most once (the client makes only one request) or at least once (the client keeps requesting until success is confirmed) is relatively straightforward. However, it's challenging to implement *exactly once* behavior. To achieve this, your clients should generate and provide an idempotency token for each request.

By using idempotency tokens, a service can distinguish between new requests and repeated ones. When a service receives a request with an idempotency token, it checks if the token has already been used. If the token has been used, the service retrieves and returns the stored response. If the token is new, the service processes the request, stores the response along with the token, and then returns the response. This mechanism makes all responses idempotent, which enhances the reliability and consistency of the distributed system.

Idempotency is also an important behavior of event-driven architectures. These architectures are typically backed by a message queue such as Amazon SQS, Amazon MQ, Amazon Kinesis Streams, or Amazon Managed Streaming for Apache Kafka (MSK). In some circumstances, a message that was published only once may be accidentally delivered more than once. When a publisher generates and includes idempotency tokens in messages, it requests that the processing of any duplicate message received doesn't result in a repeated action for the same message. Consumers should keep track of each token received and ignore messages that contain duplicate tokens.

Services and consumers should also pass the received idempotency token to any downstream services that it calls. Every downstream service in the processing chain is similarly responsible for making sure that idempotency is implemented to avoid the side effect of processing a message more than once.

Implementation steps

1. **Identify idempotent operations**

   Determine which operations require idempotency. These typically include POST, PUT, and DELETE HTTP methods and database insert, update, or delete operations. Operations that do not mutate state, such as read-only queries, usually do not require idempotency unless they have side effects.

2. **Use unique identifiers**

   Include a unique token in each idempotent operation request sent by the sender, either directly in the request or as part of its metadata (for example, an HTTP header). This allows the receiver to recognize and handle duplicate requests or operations. Identifiers commonly used for tokens include Universally Unique Identifiers (UUIDs) and K-Sortable Unique Identifiers (KSUIDs).

3. **Track and manage state**

   Maintain the state of each operation or request in your workload. This can be achieved by storing the idempotency token and the corresponding state (such as pending, completed, or

failed) in a database, cache, or other persistent store. This state information allows the workload to identify and handle duplicate requests or operations.

Maintain consistency and atomicity by using appropriate concurrency control mechanisms if needed, such as locks, transactions, or optimistic concurrency controls. This includes the process of recording the idempotent token and running all mutating operations associated with servicing the request. This helps prevent race conditions and verifies that idempotent operations run correctly.

Regularly remove old idempotency tokens from the datastore to manage storage and performance. If your storage system supports it, consider using expiration timestamps for data (often known as time to live, or TTL values). The likelihood of idempotency token reuse diminishes over time.

Common AWS storage options typically used for storing idempotency tokens and related state include:

- **Amazon DynamoDB**: DynamoDB is a NoSQL database service that provides low-latency performance and high availability, which makes it well-suited for the storage of idempotency-related data. The key-value and document data model of DynamoDB allows for efficient storage and retrieval of idempotency tokens and associated state information. DynamoDB can also expire idempotency tokens automatically if your application sets a TTL value when it inserts them.
- **Amazon ElastiCache**: ElastiCache can store idempotency tokens with high throughput, low latency, and at low cost. Both ElastiCache (Redis) and ElastiCache (Memcached) can also expire idempotency tokens automatically if your application sets a TTL value when it inserts them.
- **Amazon Relational Database Service (RDS):** You can use Amazon RDS to store idempotency tokens and related state information, especially if your application already uses a relational database for other purposes.
- **Amazon Simple Storage Service (S3):** Amazon S3 is a highly scalable and durable object storage service that can be used to store idempotency tokens and related metadata. The versioning capabilities of S3 can be particularly useful for maintenance of the state of idempotent operations. The choice of storage service typically depends on factors such as the volume of idempotency-related data, the required performance characteristics, the need for durability and availability, and how the idempotency mechanism integrates with the overall workload architecture.

4. **Implement idempotent operations**

Design your API and workload components to be idempotent. Incorporate idempotency checks into your workload components. Before you process a request or perform an operation, check if the unique identifier has already been processed. If it has, return the previous result instead of executing the operation again. For example, if a client sends a request to create a user, check if a user with the same unique identifier already exists. If the user exists, it should return the existing user information instead of creating a new one. Similarly, if a queue consumer receives a message with a duplicate idempotency token, the consumer should ignore the message.

Create comprehensive test suites that validate the idempotency of requests. They should cover a wide range of scenarios, such as successful requests, failed requests, and duplicate requests.

If your workload leverages AWS Lambda functions, consider Powertools for AWS Lambda. Powertools for AWS Lambda is a developer toolkit that helps implement serverless best practices and increase developer velocity when you work with AWS Lambda functions. In particular, it provides a utility to convert your Lambda functions into idempotent operations which are safe to retry.

5. **Communicate idempotency clearly**

   Document your API and workload components to clearly communicate the idempotent nature of the operations. This helps clients understand the expected behavior and how to interact with your workload reliably.

6. **Monitor and audit**

   Implement monitoring and auditing mechanisms to detect any issues related to the idempotency of responses, such as unexpected response variations or excessive duplicate request handling. This can help you detect and investigate any issues or unexpected behaviors in your workload.

Resources

Related best practices:

- REL05-BP03 Control and limit retry calls
- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP03 Send notifications (Real-time processing and alarming))
- REL08-BP02 Integrate functional testing as part of your deployment

Related documents:

- The Amazon Builders' Library: Making retries safe with idempotent APIs
- The Amazon Builders' Library: Challenges with distributed systems
- The Amazon Builders' Library: Reliability, constant work, and a good cup of coffee
- Amazon Elastic Container Service: Ensuring idempotency
- How do I make my Lambda function idempotent?
- Ensuring idempotency in Amazon EC2 API requests

Related videos:

- Building Distributed Applications with Event-driven Architecture - AWS Online Tech Talks
- AWS re:Invent 2023 - Building next-generation applications with event-driven architecture
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2018 - Close Loops and Opening Minds: How to Take Control of Systems, Big and Small ARC337 (includes loose coupling, constant work, static stability)
- AWS re:Invent 2019 - Moving to event-driven architectures (SVS308)

Related tools:

- Idempotency with AWS Lambda Powertools (Java)
- Idempotency with AWS Lambda Powertools (Python)
- AWS Lambda Powertools GitHub page

REL 5. How do you design interactions in a distributed system to mitigate or withstand failures?

Distributed systems rely on communications networks to interconnect components (such as servers or services). Your workload must operate reliably despite data loss or latency over these networks. Components of the distributed system must operate in a way that does not negatively impact other components or the workload. These best practices permit workloads to withstand stresses or failures, more quickly recover from them, and mitigate the impact of such impairments. The result is improved mean time to recovery (MTTR).

Best practices

- REL05-BP01 Implement graceful degradation to transform applicable hard dependencies into soft dependencies
- REL05-BP02 Throttle requests
- REL05-BP03 Control and limit retry calls
- REL05-BP04 Fail fast and limit queues
- REL05-BP05 Set client timeouts
- REL05-BP06 Make systems stateless where possible
- REL05-BP07 Implement emergency levers

REL05-BP01 Implement graceful degradation to transform applicable hard dependencies into soft dependencies

Application components should continue to perform their core function even if dependencies become unavailable. They might be serving slightly stale data, alternate data, or even no data. This ensures overall system function is only minimally impeded by localized failures while delivering the central business value.

Desired outcome: When a component's dependencies are unhealthy, the component itself can still function, although in a degraded manner. Failure modes of components should be seen as normal operation. Workflows should be designed in such a way that such failures do not lead to complete failure or at least to predictable and recoverable states.

Common anti-patterns:

- Not identifying the core business functionality needed. Not testing that components are functional even during dependency failures.
- Serving no data on errors or when only one out of multiple dependencies is unavailable and partial results can still be returned.
- Creating an inconsistent state when a transaction partially fails.
- Not having an alternative way to access a central parameter store.
- Invalidating or emptying local state as a result of a failed refresh without considering the consequences of doing so.

Benefits of establishing this best practice: Graceful degradation improves the availability of the system as a whole and maintains the functionality of the most important functions even during failures.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implementing graceful degradation helps minimize the impact of dependency failures on component function. Ideally, a component detects dependency failures and works around them in a way that minimally impacts other components or customers.

Architecting for graceful degradation means considering potential failure modes during dependency design. For each failure mode, have a way to deliver most or at least the most critical functionality of the component to callers or customers. These considerations can become additional requirements that can be tested and verified. Ideally, a component is able to perform its core function in an acceptable manner even when one or multiple dependencies fail.

This is as much a business discussion as a technical one. All business requirements are important and should be fulfilled if possible. However, it still makes sense to ask what should happen when not all of them can be fulfilled. A system can be designed to be available and consistent, but under circumstances where one requirement must be dropped, which one is more important? For payment processing, it might be consistency. For a real-time application, it might be availability. For a customer facing website, the answer may depend on customer expectations.

What this means depends on the requirements of the component and what should be considered its core function. For example:

- An ecommerce website might display data from multiple different systems like personalized recommendations, highest ranked products, and status of customer orders on the landing page. When one upstream system fails, it still makes sense to display everything else instead of showing an error page to a customer.
- A component performing batch writes can still continue processing a batch if one of the individual operations fails. It should be simple to implement a retry mechanism. This can be done by returning information on which operations succeeded, which failed, and why they failed to the caller, or putting failed requests into a dead letter queue to implement asynchronous retries. Information about failed operations should be logged as well.
- A system that processes transactions must verify that either all or no individual updates are executed. For distributed transactions, the saga pattern can be used to roll back previous

operations in case a later operation of the same transaction fails. Here, the core function is maintaining consistency.

- Time critical systems should be able to deal with dependencies not responding in a timely manner. In these cases, the circuit breaker pattern can be used. When responses from a dependency start timing out, the system can switch to a closed state where no additional call are made.
- An application may read parameters from a parameter store. It can be useful to create container images with a default set of parameters and use these in case the parameter store is unavailable.

Note that the pathways taken in case of component failure need to be tested and should be significantly simpler than the primary pathway. Generally, fallback strategies should be avoided.

Implementation steps

Identify external and internal dependencies. Consider what kinds of failures can occur in them. Think about ways that minimize negative impact on upstream and downstream systems and customers during those failures.

The following is a list of dependencies and how to degrade gracefully when they fail:

1. **Partial failure of dependencies:** A component may make multiple requests to downstream systems, either as multiple requests to one system or one request to multiple systems each. Depending on the business context, different ways of handling for this may be appropriate (for more detail, see previous examples in Implementation guidance).
2. **A downstream system is unable to process requests due to high load:** If requests to a downstream system are consistently failing, it does not make sense to continue retrying. This may create additional load on an already overloaded system and make recovery more difficult. The circuit breaker pattern can be utilized here, which monitors failing calls to a downstream system. If a high number of calls are failing, it will stop sending more requests to the downstream system and only occasionally let calls through to test whether the downstream system is available again.
3. **A parameter store is unavailable:** To transform a parameter store, soft dependency caching or sane defaults included in container or machine images may be used. Note that these defaults need to be kept up-to-date and included in test suites.
4. **A monitoring service or other non-functional dependency is unavailable:** If a component is intermittently unable to send logs, metrics, or traces to a central monitoring service, it is often best to still execute business functions as usual. Silently not logging or pushing metrics for a

long time is often not acceptable. Also, some use cases may require complete auditing entries to fulfill compliance requirements.

5. **A primary instance of a relational database may be unavailable:** Amazon Relational Database Service, like almost all relational databases, can only have one primary writer instance. This creates a single point of failure for write workloads and makes scaling more difficult. This can partially be mitigated by using a Multi-AZ configuration for high availability or Amazon Aurora Serverless for better scaling. For very high availability requirements, it can make sense to not rely on the primary writer at all. For queries that only read, read replicas can be used, which provide redundancy and the ability to scale out, not just up. Writes can be buffered, for example in an Amazon Simple Queue Service queue, so that write requests from customers can still be accepted even if the primary is temporarily unavailable.

Resources

Related documents:

- Amazon API Gateway: Throttle API Requests for Better Throughput
- CircuitBreaker (summarizes Circuit Breaker from “Release It!” book)
- Error Retries and Exponential Backoff in AWS
- Michael Nygard “Release It! Design and Deploy Production-Ready Software”
- The Amazon Builders' Library: Avoiding fallback in distributed systems
- The Amazon Builders' Library: Avoiding insurmountable queue backlogs
- The Amazon Builders' Library: Caching challenges and strategies
- The Amazon Builders' Library: Timeouts, retries, and backoff with jitter

Related videos:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders’ Library (DOP328)

REL05-BP02 Throttle requests

Throttle requests to mitigate resource exhaustion due to unexpected increases in demand. Requests below throttling rates are processed while those over the defined limit are rejected with a return message indicating the request was throttled.

Desired outcome: Large volume spikes either from sudden customer traffic increases, flooding attacks, or retry storms are mitigated by request throttling, allowing workloads to continue normal processing of supported request volume.

Common anti-patterns:

- API endpoint throttles are not implemented or are left at default values without considering expected volumes.
- API endpoints are not load tested or throttling limits are not tested.
- Throttling request rates without considering request size or complexity.
- Testing maximum request rates or maximum request size, but not testing both together.
- Resources are not provisioned to the same limits established in testing.
- Usage plans have not been configured or considered for application to application (A2A) API consumers.
- Queue consumers that horizontally scale do not have maximum concurrency settings configured.
- Rate limiting on a per IP address basis has not been implemented.

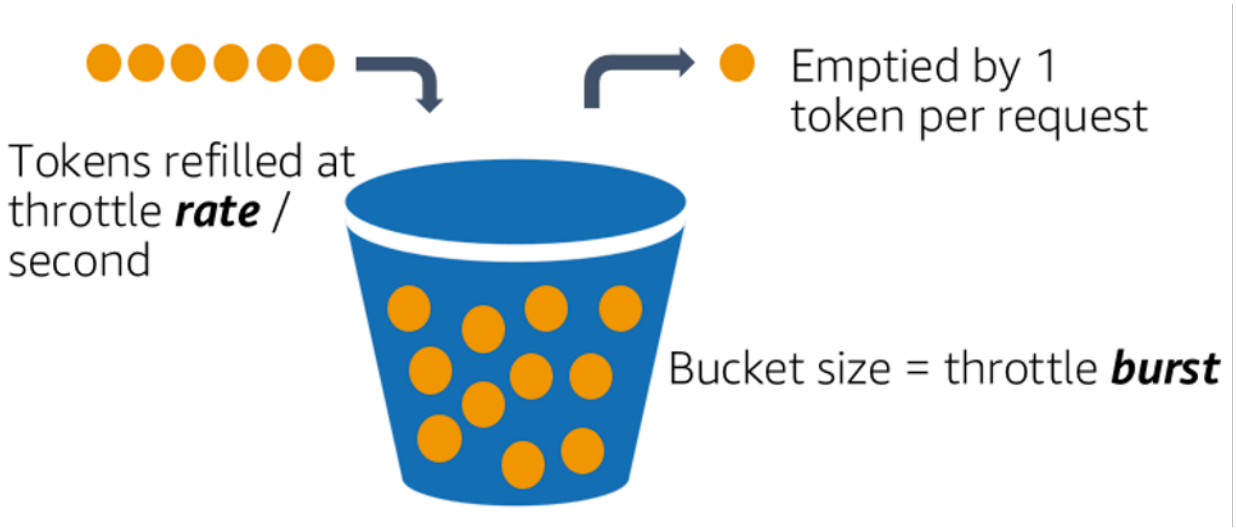
Benefits of establishing this best practice: Workloads that set throttle limits are able to operate normally and process accepted request load successfully under unexpected volume spikes. Sudden or sustained spikes of requests to APIs and queues are throttled and do not exhaust request processing resources. Rate limits throttle individual requestors so that high volumes of traffic from a single IP address or API consumer will not exhaust resources impact other consumers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Services should be designed to process a known capacity of requests; this capacity can be established through load testing. If request arrival rates exceed limits, the appropriate response signals that a request has been throttled. This allows the consumer to handle the error and retry later.

When your service requires a throttling implementation, consider implementing the token bucket algorithm, where a token counts for a request. Tokens are refilled at a throttle rate per second and emptied asynchronously by one token per request.



The token bucket algorithm.

Amazon API Gateway implements the token bucket algorithm according to account and region limits and can be configured per-client with usage plans. Additionally, Amazon Simple Queue Service (Amazon SQS) and Amazon Kinesis can buffer requests to smooth out the request rate, and allow higher throttling rates for requests that can be addressed. Finally, you can implement rate limiting with AWS WAF to throttle specific API consumers that generate unusually high load.

Implementation steps

You can configure API Gateway with throttling limits for your APIs and return `429 Too Many Requests` errors when limits are exceeded. You can use AWS WAF with your AWS AppSync and API Gateway endpoints to enable rate limiting on a per IP address basis. Additionally, where your system can tolerate asynchronous processing, you can put messages into a queue or stream to speed up responses to service clients, which allows you to burst to higher throttle rates.

With asynchronous processing, when you've configured Amazon SQS as an event source for AWS Lambda, you can configure maximum concurrency to avoid high event rates from consuming available account concurrent execution quota needed for other services in your workload or account.

While API Gateway provides a managed implementation of the token bucket, in cases where you cannot use API Gateway, you can take advantage of language specific open-source implementations (see related examples in Resources) of the token bucket for your services.

- Understand and configure API Gateway throttling limits at the account level per region, API per stage, and API key per usage plan levels.

- Apply AWS WAF rate limiting rules to API Gateway and AWS AppSync endpoints to protect against floods and block malicious IPs. Rate limiting rules can also be configured on AWS AppSync API keys for A2A consumers.
- Consider whether you require more throttling control than rate limiting for AWS AppSync APIs, and if so, configure an API Gateway in front of your AWS AppSync endpoint.
- When Amazon SQS queues are set up as triggers for Lambda queue consumers, set maximum concurrency to a value that processes enough to meet your service level objectives but does not consume concurrency limits impacting other Lambda functions. Consider setting reserved concurrency on other Lambda functions in the same account and region when you consume queues with Lambda.
- Use API Gateway with native service integrations to Amazon SQS or Kinesis to buffer requests.
- If you cannot use API Gateway, look at language specific libraries to implement the token bucket algorithm for your workload. Check the examples section and do your own research to find a suitable library.
- Test limits that you plan to set, or that you plan to allow to be increased, and document the tested limits.
- Do not increase limits beyond what you establish in testing. When increasing a limit, verify that provisioned resources are already equivalent to or greater than those in test scenarios before applying the increase.

Resources

Related best practices:

- REL04-BP03 Do constant work
- REL05-BP03 Control and limit retry calls

Related documents:

- Amazon API Gateway: Throttle API Requests for Better Throughput
- AWS WAF: Rate-based rule statement
- Introducing maximum concurrency of AWS Lambda when using Amazon SQS as an event source
- AWS Lambda: Maximum Concurrency

Related examples:

- The three most important AWS WAF rate-based rules
- Java Bucket4j
- Python token-bucket
- Node token-bucket
- .NET System Threading Rate Limiting

Related videos:

- Implementing GraphQL API security best practices with AWS AppSync

Related tools:

- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon Kinesis
- AWS WAF
- Virtual Waiting Room on AWS

REL05-BP03 Control and limit retry calls

Use exponential backoff to retry requests at progressively longer intervals between each retry. Introduce jitter between retries to randomize retry intervals. Limit the maximum number of retries.

Desired outcome: Typical components in a distributed software system include servers, load balancers, databases, and DNS servers. During normal operation, these components can respond to requests with errors that are temporary or limited, and also errors that would be persistent regardless of retries. When clients make requests to services, the requests consume resources including memory, threads, connections, ports, or any other limited resources. Controlling and limiting retries is a strategy to release and minimize consumption of resources so that system components under strain are not overwhelmed.

When client requests time out or receive error responses, they should determine whether or not to retry. If they do retry, they do so with exponential backoff with jitter and a maximum retry value. As a result, backend services and processes are given relief from load and time to self-heal, resulting in faster recovery and successful request servicing.

Common anti-patterns:

- Implementing retries without adding exponential backoff, jitter, and maximum retry values. Backoff and jitter help avoid artificial traffic spikes due to unintentionally coordinated retries at common intervals.
- Implementing retries without testing their effects or assuming retries are already built into an SDK without testing retry scenarios.
- Failing to understand published error codes from dependencies, leading to retrying all errors, including those with a clear cause that indicates lack of permission, configuration error, or another condition that predictably will not resolve without manual intervention.
- Not addressing observability practices, including monitoring and alerting on repeated service failures so that underlying issues are made known and can be addressed.
- Developing custom retry mechanisms when built-in or third-party retry capabilities suffice.
- Retrying at multiple layers of your application stack in a manner which compounds retry attempts further consuming resources in a retry storm. Be sure to understand how these errors affect your application the dependencies you rely on, then implement retries at only one level.
- Retrying service calls that are not idempotent, causing unexpected side effects like duplicated results.

Benefits of establishing this best practice: Retries help clients acquire desired results when requests fail but also consume more of a server's time to get the successful responses they want. When failures are rare or transient, retries work well. When failures are caused by resource overload, retries can make things worse. Adding exponential backoff with jitter to client retries allows servers to recover when failures are caused by resource overload. Jitter avoids alignment of requests into spikes, and backoff diminishes load escalation caused by adding retries to normal request load. Finally, it's important to configure a maximum number of retries or elapsed time to avoid creating backlogs that produce metastable failures.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Control and limit retry calls. Use exponential backoff to retry after progressively longer intervals. Introduce jitter to randomize retry intervals and limit the maximum number of retries.

Some AWS SDKs implement retries and exponential backoff by default. Use these built-in AWS implementations where applicable in your workload. Implement similar logic in your workload

when calling services that are idempotent and where retries improve your client availability. Decide what the timeouts are and when to stop retrying based on your use case. Build and exercise testing scenarios for those retry use cases.

Implementation steps

- Determine the optimal layer in your application stack to implement retries for the services your application relies on.
- Be aware of existing SDKs that implement proven retry strategies with exponential backoff and jitter for your language of choice, and favor these over writing your own retry implementations.
- Verify that services are idempotent before implementing retries. Once retries are implemented, be sure they are both tested and regularly exercise in production.
- When calling AWS service APIs, use the AWS SDKs and AWS CLI and understand the retry configuration options. Determine if the defaults work for your use case, test, and adjust as needed.

Resources

Related best practices:

- REL04-BP04 Make mutating operations idempotent
- REL05-BP02 Throttle requests
- REL05-BP04 Fail fast and limit queues
- REL05-BP05 Set client timeouts
- REL11-BP01 Monitor all components of the workload to detect failures

Related documents:

- Error Retries and Exponential Backoff in AWS
- The Amazon Builders' Library: Timeouts, retries, and backoff with jitter
- Exponential Backoff and Jitter
- Making retries safe with idempotent APIs

Related examples:

- Spring Retry

- Resilience4j Retry

Related videos:

- Retry, backoff, and jitter: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

Related tools:

- AWS SDKs and Tools: Retry behavior
- AWS Command Line Interface: AWS CLI retries

REL05-BP04 Fail fast and limit queues

When a service is unable to respond successfully to a request, fail fast. This allows resources associated with a request to be released, and permits a service to recover if it's running out of resources. Failing fast is a well-established software design pattern that can be leveraged to build highly reliable workloads in the cloud. Queuing is also a well-established enterprise integration pattern that can smooth load and allow clients to release resources when asynchronous processing can be tolerated. When a service is able to respond successfully under normal conditions but fails when the rate of requests is too high, use a queue to buffer requests. However, do not allow a buildup of long queue backlogs that can result in processing stale requests that a client has already given up on.

Desired outcome: When systems experience resource contention, timeouts, exceptions, or grey failures that make service level objectives unachievable, fail fast strategies allow for faster system recovery. Systems that must absorb traffic spikes and can accommodate asynchronous processing can improve reliability by allowing clients to quickly release requests by using queues to buffer requests to backend services. When buffering requests to queues, queue management strategies are implemented to avoid insurmountable backlogs.

Common anti-patterns:

- Implementing message queues but not configuring dead letter queues (DLQ) or alarms on DLQ volumes to detect when a system is in failure.
- Not measuring the age of messages in a queue, a measurement of latency to understand when queue consumers are falling behind or erroring out causing retrying.

- Not clearing backlogged messages from a queue, when there is no value in processing these messages if the business need no longer exists.
- Configuring first in first out (FIFO) queues when last in first out (LIFO) queues would better serve client needs, for example when strict ordering is not required and backlog processing is delaying all new and time sensitive requests resulting in all clients experiencing breached service levels.
- Exposing internal queues to clients instead of exposing APIs that manage work intake and place requests into internal queues.
- Combining too many work request types into a single queue which can exacerbate backlog conditions by spreading resource demand across request types.
- Processing complex and simple requests in the same queue, despite needing different monitoring, timeouts and resource allocations.
- Not validating inputs or using assertions to implement fail fast mechanisms in software that bubble up exceptions to higher level components that can handle errors gracefully.
- Not removing faulty resources from request routing, especially when failures are grey emitting both successes and failures due to crashing and restarting, intermittent dependency failure, reduced capacity, or network packet loss.

Benefits of establishing this best practice: Systems that fail fast are easier to debug and fix, and often expose issues in coding and configuration before releases are published into production. Systems that incorporate effective queueing strategies provide greater resilience and reliability to traffic spikes and intermittent system fault conditions.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Fail fast strategies can be coded into software solutions as well as configured into infrastructure. In addition to failing fast, queues are a straightforward yet powerful architectural technique to decouple system components smooth load. Amazon CloudWatch provides capabilities to monitor for and alarm on failures. Once a system is known to be failing, mitigation strategies can be invoked, including failing away from impaired resources. When systems implement queues with Amazon SQS and other queue technologies to smooth load, they must consider how to manage queue backlogs, as well as message consumption failures.

Implementation steps

- Implement programmatic assertions or specific metrics in your software and use them to explicitly alert on system issues. Amazon CloudWatch helps you create metrics and alarms based on application log pattern and SDK instrumentation.
- Use CloudWatch metrics and alarms to fail away from impaired resources that are adding latency to processing or repeatedly failing to process requests.
- Use asynchronous processing by designing APIs to accept requests and append requests to internal queues using Amazon SQS and then respond to the message-producing client with a success message so the client can release resources and move on with other work while backend queue consumers process requests.
- Measure and monitor for queue processing latency by producing a CloudWatch metric each time you take a message off a queue by comparing now to message timestamp.
- When failures prevent successful message processing or traffic spikes in volumes that cannot be processed within service level agreements, sideline older or excess traffic to a spillover queue. This allows priority processing of new work, and older work when capacity is available. This technique is an approximation of LIFO processing and allows normal system processing for all new work.
- Use dead letter or redrive queues to move messages that can't be processed out of the backlog into a location that can be researched and resolved later
- Either retry or, when tolerable, drop old messages by comparing now to the message timestamp and discarding messages that are no longer relevant to the requesting client.

Resources

Related best practices:

- REL04-BP02 Implement loosely coupled dependencies
- REL05-BP02 Throttle requests
- REL05-BP03 Control and limit retry calls
- REL06-BP02 Define and calculate metrics (Aggregation)
- REL06-BP07 Monitor end-to-end tracing of requests through your system

Related documents:

- Avoiding insurmountable queue backlogs
- Fail Fast
- How can I prevent an increasing backlog of messages in my Amazon SQS queue?
- Elastic Load Balancing: Zonal Shift
- Amazon Application Recovery Controller: Routing control for traffic failover

Related examples:

- Enterprise Integration Patterns: Dead Letter Channel

Related videos:

- AWS re:Invent 2022 - Operating highly available Multi-AZ applications

Related tools:

- Amazon SQS
- Amazon MQ
- AWS IoT Core
- Amazon CloudWatch

REL05-BP05 Set client timeouts

Set timeouts appropriately on connections and requests, verify them systematically, and do not rely on default values as they are not aware of workload specifics.

Desired outcome: Client timeouts should consider the cost to the client, server, and workload associated with waiting for requests that take abnormal amounts of time to complete. Since it is not possible to know the exact cause of any timeout, clients must use knowledge of services to develop expectations of probable causes and appropriate timeouts

Client connections time out based on configured values. After encountering a timeout, clients make decisions to back off and retry or open a circuit breaker. These patterns avoid issuing requests that may exacerbate an underlying error condition.

Common anti-patterns:

- Not being aware of system timeouts or default timeouts.
- Not being aware of normal request completion timing.
- Not being aware of possible causes for requests to take abnormally long to complete, or the costs to client, service, or workload performance associated with waiting on these completions.
- Not being aware of the probability of impaired network causing a request to fail only once timeout is reached, and the costs to client and workload performance for not adopting a shorter timeout.
- Not testing timeout scenarios both for connections and requests.
- Setting timeouts too high, which can result in long wait times and increase resource utilization.
- Setting timeouts too low, resulting in artificial failures.
- Overlooking patterns to deal with timeout errors for remote calls like circuit breakers and retries.
- Not considering monitoring for service call error rates, service level objectives for latency, and latency outliers. These metrics can provide insight to aggressive or permissive timeouts

Benefits of establishing this best practice: Remote call timeouts are configured and systems are designed to handle timeouts gracefully so that resources are conserved when remote calls respond abnormally slow and timeout errors are handled gracefully by service clients.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Set both a connection timeout and a request timeout on any service dependency call and generally on any call across processes. Many frameworks offer built-in timeout capabilities, but be careful, as some have default values that are infinite or higher than acceptable for your service goals. A value that is too high reduces the usefulness of the timeout because resources continue to be consumed while the client waits for the timeout to occur. A value that is too low can generate increased traffic on the backend and increased latency because too many requests are retried. In some cases, this can lead to complete outages because all requests are being retried.

Consider the following when determining timeout strategies:

- Requests may take longer than normal to process because of their content, impairments in a target service, or a networking partition failure.
- Requests with abnormally expensive content could consume unnecessary server and client resources. In this case, timing out these requests and not retrying can preserve resources.

Services should also protect themselves from abnormally expensive content with throttles and server-side timeouts.

- Requests that take abnormally long due to a service impairment can be timed out and retried. Consideration should be given to service costs for the request and retry, but if the cause is a localized impairment, a retry is not likely to be expensive and will reduce client resource consumption. The timeout may also release server resources depending on the nature of the impairment.
- Requests that take a long time to complete because the request or response has failed to be delivered by the network can be timed out and retried. Because the request or response was not delivered, failure would have been the outcome regardless of the length of timeout. Timing out in this case will not release server resources, but it will release client resources and improve workload performance.

Take advantage of well-established design patterns like retries and circuit breakers to handle timeouts gracefully and support fail-fast approaches. AWS SDKs and AWS CLI allow for configuration of both connection and request timeouts and for retries with exponential backoff and jitter. AWS Lambda functions support configuration of timeouts, and with AWS Step Functions, you can build low code circuit breakers that take advantage of pre-built integrations with AWS services and SDKs. AWS App Mesh Envoy provides timeout and circuit breaker capabilities.

Implementation steps

- Configure timeouts on remote service calls and take advantage of built-in language timeout features or open source timeout libraries.
- When your workload makes calls with an AWS SDK, review the documentation for language specific timeout configuration.
  - Python
  - PHP
  - .NET
  - Ruby
  - Java
  - Go
  - Node.js
  - C++

- When using AWS SDKs or AWS CLI commands in your workload, configure default timeout values by setting the AWS configuration defaults for `connectTimeoutInMillis` and `tlsNegotiationTimeoutInMillis`.
- Apply command line options `cli-connect-timeout` and `cli-read-timeout` to control one-off AWS CLI commands to AWS services.
- Monitor remote service calls for timeouts, and set alarms on persistent errors so that you can proactively handle error scenarios.
- Implement CloudWatch Metrics and CloudWatch anomaly detection on call error rates, service level objectives for latency, and latency outliers to provide insight into managing overly aggressive or permissive timeouts.
- Configure timeouts on Lambda functions.
- API Gateway clients must implement their own retries when handling timeouts. API Gateway supports a 50 millisecond to 29 second integration timeout for downstream integrations and does not retry when integration requests timeout.
- Implement the circuit breaker pattern to avoid making remote calls when they are timing out. Open the circuit to avoid failing calls and close the circuit when calls are responding normally.
- For container based workloads, review App Mesh Envoy features to leverage built in timeouts and circuit breakers.
- Use AWS Step Functions to build low code circuit breakers for remote service calls, especially where calling AWS native SDKs and supported Step Functions integrations to simplify your workload.

Resources

Related best practices:

- REL05-BP03 Control and limit retry calls
- REL05-BP04 Fail fast and limit queues
- REL06-BP07 Monitor end-to-end tracing of requests through your system

Related documents:

- AWS SDK: Retries and Timeouts
- The Amazon Builders' Library: Timeouts, retries, and backoff with jitter

- Amazon API Gateway quotas and important notes
- AWS Command Line Interface: Command line options
- AWS SDK for Java 2.x: Configure API Timeouts
- AWS Botocore using the config object and Config Reference
- AWS SDK for .NET: Retries and Timeouts
- AWS Lambda: Configuring Lambda function options

Related examples:

- Using the circuit breaker pattern with AWS Step Functions and Amazon DynamoDB
- Martin Fowler: CircuitBreaker

Related tools:

- AWS SDKs
- AWS Lambda
- Amazon SQS
- AWS Step Functions
- AWS Command Line Interface

REL05-BP06 Make systems stateless where possible

Systems should either not require state, or should offload state such that between different client requests, there is no dependence on locally stored data on disk and in memory. This allows servers to be replaced at will without causing an availability impact.

When users or services interact with an application, they often perform a series of interactions that form a session. A session is unique data for users that persists between requests while they use the application. A stateless application is an application that does not need knowledge of previous interactions and does not store session information.

Once designed to be stateless, you can then use serverless compute services, such as AWS Lambda or AWS Fargate.

In addition to server replacement, another benefit of stateless applications is that they can scale horizontally because any of the available compute resources (such as EC2 instances and AWS Lambda functions) can service any request.

Benefits of establishing this best practice: Systems that are designed to be stateless are more adaptable to horizontal scaling, making it possible to add or remove capacity based on fluctuating traffic and demand. They are also inherently resilient to failures and provide flexibility and agility in application development.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Make your applications stateless. Stateless applications allow horizontal scaling and are tolerant to the failure of an individual node. Analyze and understand the components of your application that maintain state within the architecture. This helps you assess the potential impact of transitioning to a stateless design. A stateless architecture decouples user data and offloads the session data. This provides the flexibility to scale each component independently to meet varying workload demands and optimize resource utilization.

Implementation steps

- Identify and understand the stateful components in your application.
- Decouple data by separating and managing user data from the core application logic.
  - Amazon Cognito can decouple user data from application code by using features, such as identity pools, user pools, and Amazon Cognito Sync.
  - You can use AWS Secrets Manager decouple user data by storing secrets in a secure, centralized location. This means that the application code doesn't need to store secrets, which makes it more secure.
  - Consider using Amazon S3 to store large, unstructured data, such as images and documents. Your application can retrieve this data when required, eliminating the need to store it in memory.
  - Use Amazon DynamoDB to store information such as user profiles. Your application can query this data in near-real time.
- Offload session data to a database, cache, or external files.
  - Amazon ElastiCache, Amazon DynamoDB, Amazon Elastic File System (Amazon EFS), and Amazon MemoryDB are examples of AWS services that you can use to offload session data.

- Design a stateless architecture after you identify which state and user data need to be persisted with your storage solution of choice.

Resources

Related best practices:

- REL11-BP03 Automate healing on all layers

Related documents:

- The Amazon Builders' Library: Avoiding fallback in distributed systems
- The Amazon Builders' Library: Avoiding insurmountable queue backlogs
- The Amazon Builders' Library: Caching challenges and strategies
- Best Practices for Stateless Web Tier on AWS

REL05-BP07 Implement emergency levers

Emergency levers are rapid processes that can mitigate availability impact on your workload.

Emergency levers work by disabling, throttling, or changing the behavior of components or dependencies using known and tested mechanisms. This can alleviate workload impairments caused by resource exhaustion due to unexpected increases in demand and reduce the impact of failures in non-critical components within your workload.

Desired outcome: By implementing emergency levers, you can establish known-good processes to maintain the availability of critical components in your workload. The workload should degrade gracefully and continue to perform its business-critical functions during the activation of an emergency lever. For more detail on graceful degradation, see REL05-BP01 Implement graceful degradation to transform applicable hard dependencies into soft dependencies.

Common anti-patterns:

- Failure of non-critical dependencies impacts the availability of your core workload.
- Not testing or verifying critical component behavior during non-critical component impairment.
- No clear and deterministic criteria defined for activation or deactivation of an emergency lever.

Benefits of establishing this best practice: Implementing emergency levers can improve the availability of the critical components in your workload by providing your resolvers with established processes to respond to unexpected spikes in demand or failures of non-critical dependencies.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Identify critical components in your workload.
- Design and architect the critical components in your workload to withstand failure of non-critical components.
- Conduct testing to validate the behavior of your critical components during the failure of non-critical components.
- Define and monitor relevant metrics or triggers to initiate emergency lever procedures.
- Define the procedures (manual or automated) that comprise the emergency lever.

Implementation steps

- Identify business-critical components in your workload.
  - Each technical component in your workload should be mapped to its relevant business function and ranked as critical or non-critical. For examples of critical and non-critical functionality at Amazon, see Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second.
  - This is both a technical and business decision, and varies by organization and workload.
- Design and architect the critical components in your workload to withstand failure of non-critical components.
  - During dependency analysis, consider all potential failure modes, and verify that your emergency lever mechanisms deliver the critical functionality to downstream components.
- Conduct testing to validate the behavior of your critical components during activation of your emergency levers.
  - Avoid bimodal behavior. For more detail, see REL11-BP05 Use static stability to prevent bimodal behavior.
- Define, monitor, and alert on relevant metrics to initiate the emergency lever procedure.

  - Finding the right metrics to monitor depends on your workload. Some example metrics are latency or the number of failed request to a dependency.
- Define the procedures, manual or automated, that comprise the emergency lever.
  - This may include mechanisms such as load shedding, throttling requests, or implementing graceful degradation.

Resources

Related best practices:

- REL05-BP01 Implement graceful degradation to transform applicable hard dependencies into soft dependencies
- REL05-BP02 Throttle requests
- REL11-BP05 Use static stability to prevent bimodal behavior

Related documents:

- Automating safe, hands-off deployments
- Any Day Can Be Prime Day: How Amazon.com Search Uses Chaos Engineering to Handle Over 84K Requests Per Second

Related videos:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

Change management

Questions

- REL 6. How do you monitor workload resources?
- REL 7. How do you design your workload to adapt to changes in demand?
- REL 8. How do you implement change?

REL 6. How do you monitor workload resources?

Logs and metrics are powerful tools to gain insight into the health of your workload. You can configure your workload to monitor logs and metrics and send notifications when thresholds are crossed or significant events occur. Monitoring allows your workload to recognize when low-performance thresholds are crossed or failures occur, so it can recover automatically in response.

Best practices

- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP02 Define and calculate metrics (Aggregation)
- REL06-BP03 Send notifications (Real-time processing and alarming)
- REL06-BP04 Automate responses (Real-time processing and alarming)
- REL06-BP05 Analyze logs
- REL06-BP06 Regularly review monitoring scope and metrics
- REL06-BP07 Monitor end-to-end tracing of requests through your system

REL06-BP01 Monitor all components for the workload (Generation)

Monitor the components of the workload with Amazon CloudWatch or third-party tools. Monitor AWS services with AWS Health Dashboard.

All components of your workload should be monitored, including the front-end, business logic, and storage tiers. Define key metrics, describe how to extract them from logs (if necessary), and set thresholds for invoking corresponding alarm events. Ensure metrics are relevant to the key performance indicators (KPIs) of your workload, and use metrics and logs to identify early warning signs of service degradation. For example, a metric related to business outcomes such as the number of orders successfully processed per minute, can indicate workload issues faster than technical metric, such as CPU Utilization. Use AWS Health Dashboard for a personalized view into the performance and availability of the AWS services underlying your AWS resources.

Monitoring in the cloud offers new opportunities. Most cloud providers have developed customizable hooks and can deliver insights to help you monitor multiple layers of your workload. AWS services such as Amazon CloudWatch apply statistical and machine learning algorithms to continually analyze metrics of systems and applications, determine normal baselines, and surface anomalies with minimal user intervention. Anomaly detection algorithms account for the seasonality and trend changes of metrics.

AWS makes an abundance of monitoring and log information available for consumption that can be used to define workload-specific metrics, change-in-demand processes, and adopt machine learning techniques regardless of ML expertise.

In addition, monitor all of your external endpoints to ensure that they are independent of your base implementation. This active monitoring can be done with synthetic transactions (sometimes referred to as *user canaries*, but not to be confused with canary deployments) which periodically run a number of common tasks matching actions performed by clients of the workload. Keep these tasks short in duration and be sure not to overload your workload during testing. Amazon CloudWatch Synthetics allows you to create synthetic canaries to monitor your endpoints and APIs. You can also combine the synthetic canary client nodes with AWS X-Ray console to pinpoint which synthetic canaries are experiencing issues with errors, faults, or throttling rates for the selected time frame.

Desired Outcome:

Collect and use critical metrics from all components of the workload to ensure workload reliability and optimal user experience. Detecting that a workload is not achieving business outcomes allows you to quickly declare a disaster and recover from an incident.

Common anti-patterns:

- Only monitoring external interfaces to your workload.
- Not generating any workload-specific metrics and only relying on metrics provided to you by the AWS services your workload uses.
- Only using technical metrics in your workload and not monitoring any metrics related to non-technical KPIs the workload contributes to.
- Relying on production traffic and simple health checks to monitor and evaluate workload state.

Benefits of establishing this best practice: Monitoring at all tiers in your workload allows you to more rapidly anticipate and resolve problems in the components that comprise the workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

1. **Turn on logging where available.** Monitoring data should be obtained from all components of the workloads. Turn on additional logging, such as S3 Access Logs, and permit your workload

to log workload specific data. Collect metrics for CPU, network I/O, and disk I/O averages from services such as Amazon ECS, Amazon EKS, Amazon EC2, Elastic Load Balancing, AWS Auto Scaling, and Amazon EMR. See AWS Services That Publish CloudWatch Metrics for a list of AWS services that publish metrics to CloudWatch.

2. **Review all default metrics and explore any data collection gaps.** Every service generates default metrics. Collecting default metrics allows you to better understand the dependencies between workload components, and how component reliability and performance affect the workload. You can also create and publish your own metrics to CloudWatch using the AWS CLI or an API.
3. **Evaluate all the metrics to decide which ones to alert on for each AWS service in your workload.** You may choose to select a subset of metrics that have a major impact on workload reliability. Focusing on critical metrics and threshold allows you to refine the number of alerts and can help minimize false-positives.
4. **Define alerts and the recovery process for your workload after the alert is invoked.** Defining alerts allows you to quickly notify, escalate, and follow steps necessary to recover from an incident and meet your prescribed Recovery Time Objective (RTO). You can use Amazon CloudWatch Alarms to invoke automated workflows and initiate recovery procedures based on defined thresholds.
5. **Explore use of synthetic transactions to collect relevant data about workloads state.** Synthetic monitoring follows the same routes and perform the same actions as a customer, which makes it possible for you to continually verify your customer experience even when you don't have any customer traffic on your workloads. By using synthetic transactions, you can discover issues before your customers do.

Resources

Related best practices:

- REL11-BP03 Automate healing on all layers

Related documents:

- Getting started with your AWS Health Dashboard – Your account health
- AWS Services That Publish CloudWatch Metrics
- Access Logs for Your Network Load Balancer
- Access logs for your application load balancer

- Accessing Amazon CloudWatch Logs for AWS Lambda
- Amazon S3 Server Access Logging
- Enable Access Logs for Your Classic Load Balancer
- Exporting log data to Amazon S3
- Install the CloudWatch agent on an Amazon EC2 instance
- Publishing Custom Metrics
- Using Amazon CloudWatch Dashboards
- Using Amazon CloudWatch Metrics
- Using Canaries (Amazon CloudWatch Synthetics)
- What are Amazon CloudWatch Logs?

User guides:

- Creating a trail
- Monitoring memory and disk metrics for Amazon EC2 Linux instances
- Using CloudWatch Logs with container instances
- VPC Flow Logs
- What is Amazon DevOps Guru?
- What is AWS X-Ray?

Related blogs:

- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray

Related examples:

- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- Observability workshop

REL06-BP02 Define and calculate metrics (Aggregation)

Collect metrics and logs from your workload components and calculate relevant aggregate metrics from them. These metrics provide broad and deep observability of your workload and can significantly improve your resilience posture.

Observability is more than just collecting metrics from workload components and being able to view and alert on them. It's about having a holistic understanding about your workload's behavior. This behavioral information comes from all components in your workloads, which includes the cloud services on which they depend, well-crafted logs, and metrics. This data gives you oversight on your workload's behavior as a whole, as well as an understanding of every component's interaction with every unit of work at a fine level of detail.

Desired outcome:

- You collect logs from your workload components and AWS service dependencies, and you publish them to a central location where they can be easily accessed and processed.
- Your logs contain high-fidelity and accurate timestamps.
- Your logs contain relevant information about the processing context, such as a trace identifier, user or account identifier, and remote IP address.
- You create aggregate metrics from your logs that represent your workload's behavior from a high-level perspective.
- You are able to query your aggregated logs to gain deep and relevant insights about your workload and identify actual and potential problems.

Common anti-patterns:

- You don't collect relevant logs or metrics from the compute instances your workloads run on or the cloud services they use.
- You overlook the collection of logs and metrics related to your business key performance indicators (KPIs).
- You analyze workload-related telemetry in isolation without aggregation and correlation.
- You allow metrics and logs to expire too quickly, which hinders trend analysis and recurring issue identification.

Benefits of establishing these best practices: You can detect more anomalies and correlate events and metrics between different components of your workload. You can create insights from your workload components based on information contained in logs that frequently aren't available in metrics alone. You can determine causes of failure more quickly by querying your logs at scale.

Level of risk exposed if these best practices are not established: High

Implementation guidance

Identify the sources of telemetry data that are relevant for your workloads and their components. This data comes not only from components that publish metrics, such as your operating system (OS) and application runtimes such as Java, but also from application and cloud service logs. For example, web servers typically log each request with detailed information such as the timestamp, processing latency, user ID, remote IP address, path, and query string. The level of detail in these logs helps you perform detailed queries and generate metrics that may not have been otherwise available.

Collect the metrics and logs using appropriate tools and processes. Logs generated by applications running on Amazon EC2 instance can be collected by an agent such as the Amazon CloudWatch Agent and published to a central storage service such as Amazon CloudWatch Logs. AWS-managed compute services such as AWS Lambda and Amazon Elastic Container Service publish logs to CloudWatch Logs for you automatically. Enable log collection for AWS storage and processing services used by your workloads such as Amazon CloudFront, Amazon S3, Elastic Load Balancing, and Amazon API Gateway.

Enrich your telemetry data with *dimensions* that can help you see behavioral patterns more clearly and isolate correlated problems to groups of related components. Once added, you can observe component behavior at a finer level of detail, detect correlated failures, and take appropriate remedial steps. Examples of useful dimensions include Availability Zone, EC2 instance ID, and container task or Pod ID.

Once you have collected the metrics and logs, you can write queries and generate aggregate metrics from them that provide useful insights into both normal and anomalous behavior. For example, you can use Amazon CloudWatch Logs Insights to derive custom metrics from your application logs, Amazon CloudWatch Metrics Insights to query your metrics at scale, Amazon CloudWatch Container Insights to collect, aggregate and summarize metrics and logs from your containerized applications and microservices, or Amazon CloudWatch Lambda Insights if you're using AWS Lambda functions. To create an aggregate error rate metric, you can increment a counter each time an error response or message is found in your component logs or calculate the aggregate value of an existing error rate metric. You can use this data to generate histograms that show *tail behavior*, such as the worst-performing requests or processes. You can also scan this data in real time for anomalous patterns using solutions such as CloudWatch Logs anomaly detection. These insights can be placed on dashboards to keep them organized according to your needs and preferences.

Querying logs can help you understand how specific requests were handled by your workload components and reveal request patterns or other context that has an impact on your workload's resilience. It can be useful to research and prepare queries in advance, based on your knowledge of how your applications and other components behave, so you can more easily run them as needed. For example, with CloudWatch Logs Insights, you can interactively search and analyze your log data stored in CloudWatch Logs. You can also use Amazon Athena to query logs from multiple sources, including many AWS services, at petabyte scale.

When you define a log retention policy, consider the value of historical logs. Historical logs can help identify long-term usage and behavioral patterns, regressions, and improvements in your workload's performance. Permanently deleted logs cannot be analyzed later. However, the value of historical logs tends to diminish over long periods of time. Choose a policy that balances your needs as appropriate and is compliant with any legal or contractual requirements you might be subject to.

Implementation steps

1. Choose collection, storage, analysis, and display mechanisms for your observability data.
2. Install and configure metric and log collectors on the appropriate components of your workload (for example, on Amazon EC2 instances and in sidecar containers). Configure these collectors to restart automatically if they unexpectedly stop. Enable disk or memory buffering for the collectors so that temporary publication failures don't impact your applications or result in lost data.
3. Enable logging on AWS services you use as a part of your workloads, and forward those logs to the storage service you selected if needed. Refer to the respective services' user or developer guides for detailed instructions.
4. Define the operational metrics relevant to your workloads that are based on your telemetry data. These could be based on direct metrics emitted from your workload components, which can include business KPI related metrics, or the results of aggregated calculations such as sums, rates, percentiles, or histograms. Calculate these metrics using your log analyzer, and place them on dashboards as appropriate.
5. Prepare appropriate log queries to analyze workload components, requests, or transaction behavior as needed.
6. Define and enable a log retention policy for your component logs. Periodically delete logs when they become older than the policy permits.

Resources

Related best practices:

- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP03 Send notifications (Real-time processing and alarming)
- REL06-BP04 Automate responses (Real-time processing and alarming)
- REL06-BP05 Analyze logs
- REL06-BP06 Regularly review monitoring scope and metrics
- REL06-BP07 Monitor end-to-end tracing of requests through your system

Related documentation:

- How Amazon CloudWatch works
- Amazon Managed Prometheus
- Amazon Managed Grafana
- Analyzing log data with CloudWatch Logs Insights
- Amazon CloudWatch Lambda Insights
- Amazon CloudWatch Container Insights
- Query your metrics with CloudWatch Metrics Insights
- AWS Distro for OpenTelemetry
- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- Searching and Filtering Log Data
- Sending Logs Directly to Amazon S3
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility

Related workshops:

- One Observability Workshop

Related tools:

- AWS Distro for OpenTelemetry (GitHub)

REL06-BP03 Send notifications (Real-time processing and alarming)

When organizations detect potential issues, they send real-time notifications and alerts to the appropriate personnel and systems in order to respond quickly and effectively to these issues.

Desired outcome: Rapid responses to operational events are possible through configuration of relevant alarms based on service and application metrics. When alarm thresholds are breached, the appropriate personnel and systems are notified so they can address underlying issues.

Common anti-patterns:

- Configuring alarms with an excessively high threshold, resulting in the failure to send vital notifications.
- Configuring alarms with a threshold that is too low, resulting in inaction on important alerts due to the noise of excessive notifications.
- Not updating alarms and their threshold when usage changes.
- For alarms best addressed through automated actions, sending the notification to personnel instead of generating the automated action results in excessive notifications being sent.

Benefits of establishing this best practice: Sending real-time notifications and alerts to the appropriate personnel and systems allows for early detection of issues and rapid responses to operational incidents.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Workloads should be equipped with real-time processing and alarming to improve the detectability of issues that could impact the availability of the application and serve as triggers for automated response. Organizations can perform real-time processing and alarming by creating alerts with defined metrics in order to receive notifications whenever significant events occur or a metric exceeds a threshold.

Amazon CloudWatch allows you to create metric and composite alarms using CloudWatch alarms based on static threshold, anomaly detection, and other criteria. For more detail on the types of alarms you can configure using CloudWatch, see the alarms section of the CloudWatch documentation.

You can construct customized views of metrics and alerts of your AWS resources for your teams using CloudWatch dashboards. The customizable home pages in the CloudWatch console allow you to monitor your resources in a single view across multiple Regions.

Alarms can perform one or more actions, like sending a notification to an Amazon SNS topic, performing an Amazon EC2 action or an Amazon EC2 Auto Scaling action, or creating an OpsItem or incident in AWS Systems Manager.

Amazon CloudWatch uses Amazon SNS to send notifications when the alarm changes state, providing message delivery from the publishers (producers) to the subscribers (consumers). For more detail on setting up Amazon SNS notifications, see Configuring Amazon SNS.

CloudWatch sends EventBridge events whenever a CloudWatch alarm is created, updated, deleted, or its state changes. You can use EventBridge with these events to create rules that perform actions, such as notifying you whenever the state of an alarm changes or automatically triggering events in your account using Systems Manager automation.

Stay informed with AWS Health. AWS Health is the authoritative source of information about the health of your AWS Cloud resources. Use AWS Health to get notified of any confirmed service events so you can quickly take steps to mitigate any impact. Create purpose-fit AWS Health event notifications to e-mail and chat channels through AWS User Notifications and integrate programmatically with your monitoring and alerting tools through Amazon EventBridge. If you use AWS Organizations, aggregate AWS Health events across accounts.

When should you use EventBridge or Amazon SNS?

Both EventBridge and Amazon SNS can be used to develop event-driven applications, and your choice will depend on your specific needs.

Amazon EventBridge is recommended when you want to build an application that reacts to events from your own applications, SaaS applications, and AWS services. EventBridge is the only event-based service that integrates directly with third-party SaaS partners. EventBridge also automatically ingests events from over 200 AWS services without requiring developers to create any resources in their account.

EventBridge uses a defined JSON-based structure for events, and helps you create rules that are applied across the entire event body to select events to forward to a target. EventBridge currently supports over 20 AWS services as targets, including AWS Lambda, Amazon SQS, Amazon SNS, Amazon Kinesis Data Streams, and Amazon Data Firehose.

Amazon SNS is recommended for applications that need high fan out (thousands or millions of endpoints). A common pattern we see is that customers use Amazon SNS as a target for their rule to filter the events that they need, and fan out to multiple endpoints.

Messages are unstructured and can be in any format. Amazon SNS supports forwarding messages to six different types of targets, including Lambda, Amazon SQS, HTTP/S endpoints, SMS, mobile push, and email. Amazon SNS typical latency is under 30 milliseconds. A wide range of AWS services send Amazon SNS messages by configuring the service to do so (more than 30, including Amazon EC2, Amazon S3, and Amazon RDS).

Implementation steps

1. Create an alarm using Amazon CloudWatch alarms.
    a. A metric alarm monitors a single CloudWatch metric or an expression dependent on CloudWatch metrics. The alarm initiates one or more actions based on the value of the metric or expression in comparison to a threshold over a number of time intervals. The action may consist of sending a notification to an Amazon SNS topic, performing an Amazon EC2 action or an Amazon EC2 Auto Scaling action, or creating an OpsItem or incident in AWS Systems Manager.
    b. A composite alarm consists of a rule expression that considers the alarm conditions of other alarms you've created. The composite alarm only enters alarm state if all rule conditions are met. The alarms specified in the rule expression of a composite alarm can include metric alarms and additional composite alarms. Composite alarms can send Amazon SNS notifications when their state changes and can create Systems Manager OpsItems or incidents when they enter the alarm state, but they cannot perform Amazon EC2 or Auto Scaling actions.
2. Set up Amazon SNS notifications. When creating a CloudWatch alarm, you can include an Amazon SNS topic to send a notification when the alarm changes state.
3. Create rules in EventBridge that matches specified CloudWatch alarms. Each rule supports multiple targets, including Lambda functions. For example, you can define an alarm that initiates when available disk space is running low, which triggers a Lambda function through an EventBridge rule, to clean up the space. For more detail on EventBridge targets, see EventBridge targets.

Resources

Related Well-Architected best practices:

- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP02 Define and calculate metrics (Aggregation)
- REL12-BP01 Use playbooks to investigate failures

Related documents:

- Amazon CloudWatch
- CloudWatch Logs insights
- Using Amazon CloudWatch alarms
- Using Amazon CloudWatch dashboards
- Using Amazon CloudWatch metrics
- Setting up Amazon SNS notifications
- CloudWatch anomaly detection
- CloudWatch Logs data protection
- Amazon EventBridge
- Amazon Simple Notification Service

Related videos:

- reinvent 2022 observability videos
- AWS re:Invent 2022 - Observability best practices at Amazon

Related examples:

- One Observability Workshop
- Amazon EventBridge to AWS Lambda with feedback control by Amazon CloudWatch Alarms

REL06-BP04 Automate responses (Real-time processing and alarming)

Use automation to take action when an event is detected, for example, to replace failed components.

Automated real-time processing of alarms is implemented so that systems can take quick corrective action and attempt to prevent failures or degraded service when alarms are triggered.

Automated responses to alarms could include the replacement of failing components, the adjustment of compute capacity, the redirection of traffic to healthy hosts, availability zones, or other regions, and the notification of operators.

Desired outcome: Real-time alarms are identified, and automated processing of alarms is set up to invoke the appropriate actions taken to maintain service level objectives and service-level agreements (SLAs). Automation can range from self-healing activities of single components to full-site failover.

Common anti-patterns:

- Not having a clear inventory or catalog of key real-time alarms.
- No automated responses on critical alarms (for example, when compute is nearing exhaustion, autoscaling occurs).
- Contradictory alarm response actions.
- No standard operating procedures (SOPs) for operators to follow when they receive alert notifications.
- Not monitoring configuration changes, as undetected configuration changes can cause downtime for workloads.
- Not having a strategy to undo unintended configuration changes.

Benefits of establishing this best practice: Automating alarm processing can improve system resiliency. The system takes corrective actions automatically, reducing manual activities that allow for human, error-prone interventions. Workload operates meet availability goals, and reduces service disruption.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To effectively manage alerts and automate their response, categorize alerts based on their criticality and impact, document response procedures, and plan responses before ranking tasks.

Identify tasks requiring specific actions (often detailed in runbooks), and examine all runbooks and playbooks to determine which tasks can be automated. If actions can be defined, often they can be automated. If actions cannot be automated, document manual steps in an SOP and train operators on them. Continually challenge manual processes for automation opportunities where you can establish and maintain a plan to automate alert responses.

Implementation steps

1. **Create an inventory of alarms:** To obtain a list of all alarms, you can use the AWS CLI using the Amazon CloudWatch command `describe-alarms`. Depending upon how many alarms you have set up, you might have to use pagination to retrieve a subset of alarms for each call, or alternatively you can use the AWS SDK to obtain the alarms using an API call.
2. **Document all alarm actions:** Update a runbook with all alarms and their actions, irrespective if they are manual or automated. AWS Systems Manager provides predefined runbooks. For more information about runbooks, see Working with runbooks. For detail on how to view runbook content, see View runbook content.
3. **Set up and manage alarm actions:** For any of the alarms that require an action, specify the automated action using the CloudWatch SDK. For example, you can change the state of your Amazon EC2 instances automatically based on a CloudWatch alarm by creating and enabling actions on an alarm or disabling actions on an alarm.

   You can also use Amazon EventBridge to respond automatically to system events, such as application availability issues or resource changes. You can create rules to indicate which events you're interested in, and the actions to take when an event matches a rule. The actions that can be automatically initiated include invoking an AWS Lambda function, invoking Amazon EC2 Run Command, relaying the event to Amazon Kinesis Data Streams, and seeing Automate Amazon EC2 using EventBridge.
4. **Standard Operating Procedures (SOPs):** Based on your application components, AWS Resilience Hub recommends multiple SOP templates. You can use these SOPs to document all the processes an operator should follow in case an alert is raised. You can also construct a SOP based on Resilience Hub recommendations, where you need an Resilience Hub application with an associated resiliency policy, as well as a historic resiliency assessment against that application. The recommendations for your SOP are produced by the resiliency assessment.

   Resilience Hub works with Systems Manager to automate the steps of your SOPs by providing a number of SSM documents you can use as the basis for those SOPs. For example, Resilience Hub may recommend an SOP for adding disk space based on an existing SSM automation document.
5. **Perform automated actions using Amazon DevOps Guru:** You can use Amazon DevOps Guru to automatically monitor application resources for anomalous behavior and deliver targeted recommendations to speed up problem identification and remediation times. With DevOps Guru, you can monitor streams of operational data in near real time from multiple sources including Amazon CloudWatch metrics, AWS Config, AWS CloudFormation, and AWS X-Ray. You can

also use DevOps Guru to automatically create OpsItems in OpsCenter and send events to EventBridge for additional automation.

Resources

Related best practices:

- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP02 Define and calculate metrics (Aggregation)
- REL06-BP03 Send notifications (Real-time processing and alarming)
- REL08-BP01 Use runbooks for standard activities such as deployment

Related documents:

- AWS Systems Manager Automation
- Creating an EventBridge Rule That Triggers on an Event from an AWS Resource
- One Observability Workshop
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- What is Amazon DevOps Guru?
- Working with Automation Documents (Playbooks)

Related videos:

- AWS re:Invent 2022 - Observability best practices at Amazon
- AWS re:Invent 2020: Automate anything with AWS Systems Manager
- Introduction to AWS Resilience Hub
- Create Custom Ticket Systems for Amazon DevOps Guru Notifications
- Enable Multi-Account Insight Aggregation with Amazon DevOps Guru

Related examples:

- Amazon CloudWatch and Systems Manager Workshop

REL06-BP05 Analyze logs

Collect log files and metrics histories and analyze these for broader trends and workload insights.

Amazon CloudWatch Logs Insights supports a simple yet powerful query language that you can use to analyze log data. Amazon CloudWatch Logs also supports subscriptions that allow data to flow seamlessly to Amazon S3 where you can use or Amazon Athena to query the data. It also supports queries on a large array of formats. See Supported SerDes and Data Formats in the Amazon Athena User Guide for more information. For analysis of huge log file sets, you can run an Amazon EMR cluster to run petabyte-scale analyses.

There are a number of tools provided by AWS Partners and third parties that allow for aggregation, processing, storage, and analytics. These tools include New Relic, Splunk, Loggly, Logstash, CloudHealth, and Nagios. However, outside generation of system and application logs is unique to each cloud provider, and often unique to each service.

An often-overlooked part of the monitoring process is data management. You need to determine the retention requirements for monitoring data, and then apply lifecycle policies accordingly. Amazon S3 supports lifecycle management at the S3 bucket level. This lifecycle management can be applied differently to different paths in the bucket. Toward the end of the lifecycle, you can transition data to Amazon Glacier for long-term storage, and then expiration after the end of the retention period is reached. The S3 Intelligent-Tiering storage class is designed to optimize costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- CloudWatch Logs Insights allows you to interactively search and analyze your log data in Amazon CloudWatch Logs.
  - Analyzing Log Data with CloudWatch Logs Insights
  - Amazon CloudWatch Logs Insights Sample Queries
- Use Amazon CloudWatch Logs to send logs to Amazon S3 where you can use or Amazon Athena to query the data.
  - How do I analyze my Amazon S3 server access logs using Athena?
    - Create an S3 lifecycle policy for your server access logs bucket. Configure the lifecycle policy to periodically remove log files. Doing so reduces the amount of data that Athena analyzes for each query.

- How Do I Create a Lifecycle Policy for an S3 Bucket?

Resources

Related documents:

- Amazon CloudWatch Logs Insights Sample Queries
- Analyzing Log Data with CloudWatch Logs Insights
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- How Do I Create a Lifecycle Policy for an S3 Bucket?
- How do I analyze my Amazon S3 server access logs using Athena?
- One Observability Workshop
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility

REL06-BP06 Regularly review monitoring scope and metrics

Frequently review how workload monitoring is implemented, and update it as your workload and its architecture evolves. Regular audits of your monitoring helps reduce the risk of missed or overlooked trouble indicators and further helps your workload meet its availability goals.

Effective monitoring is anchored in key business metrics, which evolve as your business priorities change. Your monitoring review process should emphasize service-level indicators (SLIs) and incorporate insights from your infrastructure, applications, clients, and users.

Desired outcome: You have an effective monitoring strategy that is regularly reviewed and updated periodically, as well as after any significant events or changes. You verify that key application health indicators are still relevant as your workload and business requirements evolve.

Common anti-patterns:

- You collect only default metrics.
- You set up a monitoring strategy, but you never review it.
- You don't discuss monitoring when major changes are deployed.
- You trust outdated metrics to determine workload health.
- Your operations teams are overwhelmed with false-positive alerts due to outdated metrics and thresholds.

- You lack observability of application components that are not being monitored.
- You focus only on low-level technical metrics and excluding business metrics in your monitoring.

Benefits of establishing this best practice: When you regularly review your monitoring, you can anticipate potential problems and verify that you are capable of detecting them. It also allows you to uncover blind spots that you might have missed during earlier reviews, which further improves your ability to detect issues.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Review monitoring metrics and scope during your operational readiness review (ORR) process. Perform periodic operational readiness reviews on a consistent schedule to evaluate whether there are any gaps between your current workload and the monitoring you have configured. Establish a regular cadence for operational performance reviews and knowledge sharing to enhance your ability to achieve higher performance from your operational teams. Validate whether existing alert thresholds are still adequate, and check for situations where operational teams are receiving false-positive alerts or not monitoring aspects of the application that should be monitored.

The Resilience Analysis Framework provides useful guidance that can help you navigate the process. The focus of the framework is to identify potential failure modes and the preventive and corrective controls you can use to mitigate their impact. This knowledge can help you identify the right metrics and events to monitor and alert upon.

Implementation steps

1. Schedule and conduct regular reviews of the workload dashboards. You may have different cadences for the depth at which you inspect.
2. Inspect for trends in the metrics. Compare the metric values to historic values to see if there are trends that may indicate that something that needs investigation. Examples of this include increased latency, decreased primary business function, and increased failure responses.
3. Inspect for outliers and anomalies in your metrics, which can be masked by averages or medians. Look at the highest and lowest values during the time frame, and investigate the causes of observations that are far outside of normal bounds. As you continue to remove these causes, you can tighten your expected metric bounds in response to the improved consistency of your workload performance.

4. Look for sharp changes in behavior. An immediate change in quantity or direction of a metric may indicate that there has been a change in the application or external factors that you may need to add additional metrics to track.
5. Review whether the current monitoring strategy remains relevant for the application. Based on an analysis of previous incidents (or the Resilience Analysis Framework), assess if there are additional aspects of the application that should be incorporated into the monitoring scope.
6. Review your Real User Monitoring (RUM) metrics to determine whether there are any gaps in application functionality coverage.
7. Review your change management process. Update your procedures if necessary to include a monitoring analysis step that should be performed before you approve a change.
8. Implement monitoring review as part of your operational readiness review and correction of error processes.

Resources

Related best practices

- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP02 Define and calculate metrics (Aggregation)
- REL06-BP07 Monitor end-to-end tracing of requests through your system
- REL12-BP02 Perform post-incident analysis
- REL12-BP06 Conduct game days regularly

Related documents:

- Why you should develop a correction of error (COE)
- Using Amazon CloudWatch Dashboards
- Building dashboards for operational visibility
- Advanced Multi-AZ Resilience Patterns - Gray failures
- Amazon CloudWatch Logs Insights Sample Queries
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- One Observability Workshop
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility

- Using Amazon CloudWatch Dashboards
- AWS Observability Best Practices
- Resilience Analysis Framework
- Resilience Analysis Framework - Observability
- Operational Readiness Review - ORR

REL06-BP07 Monitor end-to-end tracing of requests through your system

Trace requests as they process through service components so product teams can more easily analyze and debug issues and improve performance.

Desired outcome: Workloads with comprehensive tracing across all components are easy to debug, improving mean time to resolution (MTTR) of errors and latency by simplifying root cause discovery. End-to-end tracing reduces the time it takes to discover impacted components and drill into the detailed root causes of errors or latency.

Common anti-patterns:

- Tracing is used for some components but not for all. For example, without tracing for AWS Lambda, teams might not clearly understand latency caused by cold starts in a spiky workload.
- Synthetic canaries or real-user monitoring (RUM) are not configured with tracing. Without canaries or RUM, client interaction telemetry is omitted from the trace analysis yielding an incomplete performance profile.
- Hybrid workloads include both cloud native and third party tracing tools, but steps have not been taken elect and fully integrate a single tracing solution. Based on the elected tracing solution, cloud native tracing SDKs should be used to instrument components that are not cloud native or third party tools should be configured to ingest cloud native trace telemetry.

Benefits of establishing this best practice: When development teams are alerted to issues, they can see a full picture of system component interactions, including component by component correlation to logging, performance, and failures. Because tracing makes it easy to visually identify root causes, less time is spent investigating root causes. Teams that understand component interactions in detail make better and faster decisions when resolving issues. Decisions like when to invoke disaster recovery (DR) failover or where to best implement self-healing strategies can be improved by analyzing systems traces, ultimately improving customer satisfaction with your services.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Teams that operate distributed applications can use tracing tools to establish a correlation identifier, collect traces of requests, and build service maps of connected components. All application components should be included in request traces including service clients, middleware gateways and event buses, compute components, and storage, including key value stores and databases. Include synthetic canaries and real-user monitoring in your end-to-end tracing configuration to measure remote client interactions and latency so that you can accurately evaluate your systems performance against your service level agreements and objectives.

You can use AWS X-Ray and Amazon CloudWatch Application Monitoring instrumentation services to provide a complete view of requests as they travel through your application. X-Ray collects application telemetry and allows you to visualize and filter it across payloads, functions, traces, services, APIs, and can be turned on for system components with no-code or low-code. CloudWatch application monitoring includes ServiceLens to integrate your traces with metrics, logs, and alarms. CloudWatch application monitoring also includes synthetics to monitor your endpoints and APIs, as well as real-user monitoring to instrument your web application clients.

Implementation steps

- Use AWS X-Ray on all supported native services like Amazon S3, AWS Lambda, and Amazon API Gateway. These AWS services enable X-Ray with configuration toggles using infrastructure as code, AWS SDKs, or the AWS Management Console.
- Instrument applications AWS Distro for Open Telemetry and X-Ray or third-party collection agents.
- Review the AWS X-Ray Developer Guide for programming language specific implementation. These documentation sections detail how to instrument HTTP requests, SQL queries, and other processes specific to your application programming language.
- Use X-Ray tracing for Amazon CloudWatch Synthetic Canaries and Amazon CloudWatch RUM to analyze the request path from your end user client through your downstream AWS infrastructure.
- Configure CloudWatch metrics and alarms based on resource health and canary telemetry so that teams are alerted to issues quickly, and can then deep dive into traces and service maps with ServiceLens.
- Enable X-Ray integration for third party tracing tools like Datadog, New Relic, or Dynatrace if you are using third party tools for your primary tracing solution.

Resources

Related best practices:

- REL06-BP01 Monitor all components for the workload (Generation)
- REL11-BP01 Monitor all components of the workload to detect failures

Related documents:

- What is AWS X-Ray?
- Amazon CloudWatch: Application Monitoring
- Debugging with Amazon CloudWatch Synthetics and AWS X-Ray
- The Amazon Builders' Library: Instrumenting distributed systems for operational visibility
- Integrating AWS X-Ray with other AWS services
- AWS Distro for OpenTelemetry and AWS X-Ray
- Amazon CloudWatch: Using synthetic monitoring
- Amazon CloudWatch: Use CloudWatch RUM
- Set up Amazon CloudWatch synthetics canary and Amazon CloudWatch alarm
- Availability and Beyond: Understanding and Improving the Resilience of Distributed Systems on AWS

Related examples:

- One Observability Workshop

Related videos:

- AWS re:Invent 2022 - How to monitor applications across multiple accounts
- How to Monitor your AWS Applications

Related tools:

- AWS X-Ray
- Amazon CloudWatch
- Amazon Route 53

REL 7. How do you design your workload to adapt to changes in demand?

A scalable workload provides elasticity to add or remove resources automatically so that they closely match the current demand at any given point in time.

Best practices

- REL07-BP01 Use automation when obtaining or scaling resources
- REL07-BP02 Obtain resources upon detection of impairment to a workload
- REL07-BP03 Obtain resources upon detection that more resources are needed for a workload
- REL07-BP04 Load test your workload

REL07-BP01 Use automation when obtaining or scaling resources

A cornerstone of reliability in the cloud is the programmatic definition, provisioning, and management of your infrastructure and resources. Automation helps you streamline resource provisioning, facilitate consistent and secure deployments, and scale resources across your entire infrastructure.

Desired outcome: You manage your infrastructure as code (IaC). You define and maintain your infrastructure code in version control systems (VCS). You delegate provisioning AWS resources to automated mechanisms and leverage managed services like Application Load Balancer (ALB), Network Load Balancer (NLB), and Auto Scaling groups. You provision your resources using continuous integration/continuous delivery (CI/CD) pipelines so that code changes automatically initiate resource updates, including updates to your Auto Scaling configurations.

Common anti-patterns:

- You deploy resources manually using the command line or at the AWS Management Console (also known as *click-ops*).
- You tightly couple your application components or resources, and create inflexible architectures as a result.
- You implement inflexible scaling policies that do not adapt to changing business requirements, traffic patterns, or new resource types.
- You manually estimate capacity to meet anticipated demand.

Benefits of establishing this best practice: Infrastructure as code (IaC) allows infrastructure to be defined programmatically. This helps you manage infrastructure changes through the

same software development lifecycle as application changes, which promotes consistency and repeatability and reduces the risk of manual, error-prone tasks. You can further streamline the process of provisioning and updating resources through implementing IaC with automated delivery pipelines. You can deploy infrastructure updates reliably and efficiently without the need for manual intervention. This agility is particularly important when scaling resources to meet fluctuating demands.

You can achieve dynamic, automated resource scaling in conjunction with IaC and delivery pipelines. By monitoring key metrics and applying predefined scaling policies, Auto Scaling can automatically provision or deprovision resources as needed, which improves performance and cost-efficiency. This reduces the potential for manual errors or delays in response to changes in application or workload requirements.

The combination of IaC, automated delivery pipelines, and Auto Scaling helps organizations provision, update, and scale their environments with confidence. This automation is essential to maintain a responsive, resilient, and efficiently-managed cloud infrastructure.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To set up automation with CI/CD pipelines and infrastructure as code (IaC) for your AWS architecture, choose a version control system such as Git to store your IaC templates and configuration. These templates can be written using tools such as AWS CloudFormation. To start, define your infrastructure components (such as AWS VPCs, Amazon EC2 Auto Scaling Groups, and Amazon RDS databases) within these templates.

Next, integrate these IaC templates with a CI/CD pipeline to automate the deployment process. AWS CodePipeline provides a seamless AWS-native solution, or you can use other third-party CI/CD solutions. Create a pipeline that activates when changes occur to your version control repository. Configure the pipeline to include stages that lint and validate your IaC templates, deploy the infrastructure to a staging environment, run automated tests, and finally, deploy to production. Incorporate approval steps where necessary to maintain control over changes. This automated pipeline not only speeds up deployment but also facilitates consistency and reliability across environments.

Configure Auto Scaling of resources such as Amazon EC2 instances, Amazon ECS tasks, and database replicas in your IaC to provide automatic scale-out and scale-in as needed. This approach enhances application availability and performance and optimizes cost by dynamically adjusting

resources based on demand. For a list of supported resources, see Amazon EC2 Auto Scaling and AWS Auto Scaling.

Implementation steps

1. Create and use a source code repository to store the code that controls your infrastructure configuration. Commit changes to this repository to reflect any ongoing changes you want to make.
2. Select an infrastructure as code solution such as AWS CloudFormation to keep your infrastructure up to date and detect inconsistency (drift) from your intended state.
3. Integrate your IaC platform with your CI/CD pipeline to automate deployments.
4. Determine and collect the appropriate metrics for automatic scaling of resources.
5. Configure automatic scaling of resources using scale-out and scale-in policies appropriate for your workload components. Consider using scheduled scaling for predictable usage patterns.
6. Monitor deployments to detect failures and regressions. Implement rollback mechanisms within your CI/CD platform to revert changes if necessary.

Resources

Related documents:

- AWS Auto Scaling: How Scaling Plans Work
- AWS Marketplace: products that can be used with auto scaling
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Using a load balancer with an Auto Scaling group
- What Is AWS Global Accelerator?
- What Is Amazon EC2 Auto Scaling?
- What is AWS Auto Scaling?
- What is Amazon CloudFront?
- What is Amazon Route 53?
- What is Elastic Load Balancing?
- What is a Network Load Balancer?
- What is an Application Load Balancer?

- Integrating Jenkins with AWS CodeBuild and AWS CodeDeploy
- Creating a four stage pipeline with AWS CodePipeline

Related videos:

- Back to Basics: Deploy Your Code to Amazon EC2
- AWS Supports You | Starting Your Infrastructure as Code Solution Using AWS CloudFormation Templates
- Streamline Your Software Release Process Using AWS CodePipeline
- Monitor AWS Resources Using Amazon CloudWatch Dashboards
- Create Cross Account & Cross Region CloudWatch Dashboards | Amazon Web Services

REL07-BP02 Obtain resources upon detection of impairment to a workload

Scale resources reactively when necessary if availability is impacted, to restore workload availability.

You first must configure health checks and the criteria on these checks to indicate when availability is impacted by lack of resources. Then, either notify the appropriate personnel to manually scale the resource, or start automation to automatically scale it.

Scale can be manually adjusted for your workload (for example, changing the number of EC2 instances in an Auto Scaling group, or modifying throughput of a DynamoDB table through the AWS Management Console or AWS CLI). However, automation should be used whenever possible (refer to **Use automation when obtaining or scaling resources**).

Desired outcome: Scaling activities (either automatically or manually) are initiated to restore availability upon detection of a failure or degraded customer experience.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement observability and monitoring across all components in your workload, to monitor customer experience and detect failure. Define the procedures, manual or automated, that scale the required resources. o For more information, see REL11-BP01 Monitor all components of the workload to detect failures.

Implementation steps

- Define the procedures, manual or automated, that scale the required resources.
  - Scaling procedures depend on how the different components within your workload are designed.
  - Scaling procedures also vary depending on the underlying technology utilized.
    - Components using AWS Auto Scaling can use scaling plans to configure a set of instructions for scaling your resources. If you work with AWS CloudFormation or add tags to AWS resources, you can set up scaling plans for different sets of resources per application. Auto Scaling provides recommendations for scaling strategies customized to each resource. After you create your scaling plan, Auto Scaling combines dynamic scaling and predictive scaling methods together to support your scaling strategy. For more detail, see How scaling plans work.
    - Amazon EC2 Auto Scaling verifies that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups. You can specify the minimum and maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below or above these limits. For more detail, see What is Amazon EC2 Auto Scaling?
    - Amazon DynamoDB auto scaling uses the Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns. This allows a table or a global secondary index to increase its provisioned read and write capacity to handle sudden increases in traffic, without throttling. For more detail, see Managing throughput capacity automatically with DynamoDB auto scaling.

Resources

Related best practices:

- REL07-BP01 Use automation when obtaining or scaling resources
- REL11-BP01 Monitor all components of the workload to detect failures

Related documents:

- AWS Auto Scaling: How Scaling Plans Work
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling

- What Is Amazon EC2 Auto Scaling?

REL07-BP03 Obtain resources upon detection that more resources are needed for a workload

One of the most valuable features of cloud computing is the ability to provision resources dynamically.

In traditional on-premises compute environments, you must identify and provision enough capacity in advance to serve peak demand. This is a problem because it is expensive and because it poses risks to availability if you underestimate the workload's peak capacity needs.

In the cloud, you don't have to do this. Instead, you can provision compute, database, and other resource capacity as needed to meet current and forecasted demand. Automated solutions such as Amazon EC2 Auto Scaling and Application Auto Scaling can bring resources online for you based on metrics you specify. This can make the scaling process easier and predictable, and it can make your workload significantly more reliable by ensuring you have enough resources available at all times.

Desired outcome: You configure automatic scaling of compute and other resources to meet demand. You provide sufficient headroom in your scaling policies to allow bursts of traffic to be served while additional resources are brought online.

Common anti-patterns:

- You provision a fixed number of scalable resources.
- You choose a scaling metric that does not correlate to actual demand.
- You fail to provide enough headroom in your scaling plans to accommodate demand bursts.
- Your scaling policies add capacity too late, which leads to capacity exhaustion and degraded service while additional resources are brought online.
- You fail to correctly configure minimum and maximum resource counts, which leads to scaling failures.

Benefits of establishing this best practice: Having enough resources to meet current demand is critical to provide high availability of your workload and adhere to your defined service-level objectives (SLOs). Automatic scaling allows you to provide the right amount of compute, database, and other resources your workload needs in order to serve current and forecasted demand. You don't need to determine peak capacity needs and statically allocate resources to serve it. Instead, as demand grows, you can allocate more resources to accommodate it, and after demand falls, you can deactivate resources to reduce cost.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

First, determine whether the workload component is suitable for automatic scaling. These components are called *horizontally scalable* because they provide the same resources and behave identically. Examples of horizontally-scalable components include EC2 instances that are configured alike, Amazon Elastic Container Service (ECS) tasks, and pods running on Amazon Elastic Kubernetes Service (EKS). These compute resources are typically located behind a load balancer and are referred to as *replicas*.

Other replicated resources may include database read replicas, Amazon DynamoDB tables, and Amazon ElastiCache (Redis OSS) clusters. For a complete list of supported resources, see AWS services that you can use with Application Auto Scaling.

For container-based architectures, you may need to scale two different ways. First, you may need to scale the containers that provide horizontally-scalable services. Second, you may need to scale the compute resources to make space for new containers. Different automatic scaling mechanisms exist for each layer. To scale ECS tasks, you can use Application Auto Scaling. To scale Kubernetes pods, you can use Horizontal Pod Autoscaler (HPA) or Kubernetes Event-driven Autoscaling (KEDA). To scale the compute resources, you can use Capacity Providers for ECS, or for Kubernetes, you can use Karpenter or Cluster Autoscaler.

Next, select how you will perform automatic scaling. There are three major options: metric-based scaling, scheduled scaling, and predictive scaling.

Metric-based scaling

Metric-based scaling provisions resources based on the value of one or more *scaling metrics*. A scaling metric is one that corresponds to your workload's demand. A good way to determine appropriate scaling metrics is to perform load testing in a non-production environment. During your load tests, keep the number of scalable resources fixed, and slowly increase demand (for example, throughput, concurrency, or simulated users). Then look for metrics that increase (or decrease) as demand grows, and conversely decrease (or increase) as demand falls. Typical scaling metrics include CPU utilization, work queue depth (such as an Amazon SQS queue), number of active users, and network throughput.

> **Note**
>
> AWS has observed that with most applications, memory utilization increases as the application warms up and then reaches a steady value. When demand decreases, memory utilization typically remains elevated rather than decreasing in parallel. Because memory utilization does not correspond to demand in both directions–that is, growing and falling with demand–consider carefully before you select this metric for automatic scaling.

Metric-based scaling is a *latent operation*. It can take several minutes for utilization metrics to propagate to auto scaling mechanisms, and these mechanisms typically wait for a clear signal of increased demand before reacting. Then, as the auto scaler creates new resources, it can take additional time for them to come to full service. Because of this, it is important to not set your scaling metric targets too close to full utilization (for example, 90% CPU utilization). Doing so risks exhausting existing resource capacity before additional capacity can come online. Typical resource utilization targets can range between 50-70% for optimum availability, depending on demand patterns and time required to provision additional resources.

Scheduled scaling

Scheduled scaling provisions or removes resources based on the calendar or time of day. It is frequently used for workloads that have predictable demand, such as peak utilization during weekday business hours or sales events. Both Amazon EC2 Auto Scaling and Application Auto Scaling support scheduled scaling. KEDA's cron scaler supports scheduled scaling of Kubernetes pods.

Predictive scaling

Predictive scaling uses machine learning to automatically scale resources based on anticipated demand. Predictive scaling analyzes the historical value of a utilization metric you provide and continuously predicts its future value. The predicted value is then used to scale the resource up or down. Amazon EC2 Auto Scaling can perform predictive scaling.

Implementation steps

1. Determine whether the workload component is suitable for automatic scaling.
2. Determine what kind of scaling mechanism is most appropriate for the workload: metric-based scaling, scheduled scaling, or predictive scaling.

3. Select the appropriate automatic scaling mechanism for the component. For Amazon EC2 instances, use Amazon EC2 Auto Scaling. For other AWS services, use Application Auto Scaling. For Kubernetes pods (such as those running in an Amazon EKS cluster), consider Horizontal Pod Autoscaler (HPA) or Kubernetes Event-driven Autoscaling (KEDA). For Kubernetes or EKS nodes, consider Karpenter and Cluster Auto Scaler (CAS).
4. For metric or scheduled scaling, conduct load testing to determine the appropriate scaling metrics and target values for your workload. For scheduled scaling, determine the number of resources needed at the dates and times you select. Determine the maximum number of resources needed to serve expected peak traffic.
5. Configure the auto scaler based on the information collected above. Consult the auto scaling service's documentation for details. Verify that the maximum and minimum scaling limits are configured correctly.
6. Verify the scaling configuration is working as expected. Perform load testing in a non-production environment and observe how the system reacts, and adjust as needed. When enabling auto scaling in production, configure appropriate alarms to notify you of any unexpected behavior.

Resources

Related documents:

- What Is Amazon EC2 Auto Scaling?
- AWS Prescriptive Guidance: Load testing applications
- AWS Marketplace: products that can be used with auto scaling
- Managing Throughput Capacity Automatically with DynamoDB Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Telling Stories About Little's Law

REL07-BP04 Load test your workload

Adopt a load testing methodology to measure if scaling activity meets workload requirements.

It's important to perform sustained load testing. Load tests should discover the breaking point and test the performance of your workload. AWS makes it easy to set up temporary testing environments that model the scale of your production workload. In the cloud, you can create a

production-scale test environment on demand, complete your testing, and then decommission the resources. Because you only pay for the test environment when it's running, you can simulate your live environment for a fraction of the cost of testing on premises.

Load testing in production should also be considered as part of game days where the production system is stressed, during hours of lower customer usage, with all personnel on hand to interpret results and address any problems that arise.

Common anti-patterns:

- Performing load testing on deployments that are not the same configuration as your production.
- Performing load testing only on individual pieces of your workload, and not on the entire workload.
- Performing load testing with a subset of requests and not a representative set of actual requests.
- Performing load testing to a small safety factor above expected load.

Benefits of establishing this best practice: You know what components in your architecture fail under load and be able to identify what metrics to watch to indicate that you are approaching that load in time to address the problem, preventing the impact of that failure.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

- Perform load testing to identify which aspect of your workload indicates that you must add or remove capacity. Load testing should have representative traffic similar to what you receive in production. Increase the load while watching the metrics you have instrumented to determine which metric indicates when you must add or remove resources.
  - Distributed Load Testing on AWS: simulate thousands of connected users
    - Identify the mix of requests. You may have varied mixes of requests, so you should look at various time frames when identifying the mix of traffic.
    - Implement a load driver. You can use custom code, open source, or commercial software to implement a load driver.
    - Load test initially using small capacity. You see some immediate effects by driving load onto a lesser capacity, possibly as small as one instance or container.
    - Load test against larger capacity. The effects will be different on a distributed load, so you must test against as close to a product environment as possible.

Resources

Related documents:

- Distributed Load Testing on AWS: simulate thousands of connected users
- Load testing applications

Related videos:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing

REL 8. How do you implement change?

Controlled changes are necessary to deploy new functionality, and to verify that the workloads and the operating environment are running known software and can be patched or replaced in a predictable manner. If these changes are uncontrolled, then it makes it difficult to predict the effect of these changes, or to address issues that arise because of them.

Best practices

- REL08-BP01 Use runbooks for standard activities such as deployment
- REL08-BP02 Integrate functional testing as part of your deployment
- REL08-BP03 Integrate resiliency testing as part of your deployment
- REL08-BP04 Deploy using immutable infrastructure
- REL08-BP05 Deploy changes with automation

REL08-BP01 Use runbooks for standard activities such as deployment

Runbooks are the predefined procedures to achieve specific outcomes. Use runbooks to perform standard activities, whether done manually or automatically. Examples include deploying a workload, patching a workload, or making DNS modifications.

For example, put processes in place to ensure rollback safety during deployments. Ensuring that you can roll back a deployment without any disruption for your customers is critical in making a service reliable.

For runbook procedures, start with a valid effective manual process, implement it in code, and invoke it to automatically run where appropriate.

Even for sophisticated workloads that are highly automated, runbooks are still useful for running game days or meeting rigorous reporting and auditing requirements.

Note that playbooks are used in response to specific incidents, and runbooks are used to achieve specific outcomes. Often, runbooks are for routine activities, while playbooks are used for responding to non-routine events.

Common anti-patterns:

- Performing unplanned changes to configuration in production.
- Skipping steps in your plan to deploy faster, resulting in a failed deployment.
- Making changes without testing the reversal of the change.

Benefits of establishing this best practice: Effective change planning increases your ability to successfully run the change because you are aware of all the systems impacted. Validating your change in test environments increases your confidence.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Provide consistent and prompt responses to well-understood events by documenting procedures in runbooks.
- Use the principle of infrastructure as code to define your infrastructure. By using AWS CloudFormation (or a trusted third party) to define your infrastructure, you can use version control software to version and track changes.
  - Use AWS CloudFormation (or a trusted third-party provider) to define your infrastructure.
    - What is AWS CloudFormation?
  - Create templates that are singular and decoupled, using good software design principles.
    - Determine the permissions, templates, and responsible parties for implementation.
      - Controlling access with AWS Identity and Access Management
    - Use a hosted source code management system based on a popular technology such as Git to store your source code and infrastructure as code (IaC) configuration.

Resources

Related documents:

- APN Partner: partners that can help you create automated deployment solutions
- AWS Marketplace: products that can be used to automate your deployments
- What is AWS CloudFormation?

Related examples:

- Automating operations with Playbooks and Runbooks

REL08-BP02 Integrate functional testing as part of your deployment

Use techniques such as unit tests and integration tests that validate required functionality.

Unit testing is the process where you test the smallest functional unit of code to validate its behavior. Integration testing seeks to validate that each application feature works according to the software requirements. While unit tests focus on testing part of an application in isolation, integration tests consider side effects (for example, the effect of data being changed through a mutation operation). In either case, tests should be integrated into a deployment pipeline, and if success criteria are not met, the pipeline is halted or rolled back. These tests are run in a pre-production environment, which is staged prior to production in the pipeline.

You achieve the best outcomes when these tests are run automatically as part of build and deployment actions. For instance, with AWS CodePipeline, developers commit changes to a source repository where CodePipeline automatically detects the changes. The application is built, and unit tests are run. After the unit tests have passed, the built code is deployed to staging servers for testing. From the staging server, CodePipeline runs more tests, such as integration or load tests. Upon the successful completion of those tests, CodePipeline deploys the tested and approved code to production instances.

Desired outcome: You use automation to perform unit and integration tests to validate that your code behaves as expected. These tests are integrated into the deployment process, and a test failure aborts the deployment.

Common anti-patterns:

- You ignore or bypass test failures and plans during the deployment process in order to accelerate the deployment timeline.
- You manually perform tests outside the deployment pipeline.
- You skip testing steps in the automation through manual emergency workflows.

- You run automated tests in an environment that does not closely resemble the production environment.
- You build a test suite that is insufficiently flexible and is difficult to maintain, update, or scale as the application evolves.

Benefits of establishing this best practice: Automated testing during the deployment process catches issues early, which reduces the risk of a release to production with bugs or unexpected behavior. Unit tests validate the code behaves as desired and API contracts are honored. Integration tests validate that the system operates according to specified requirements. These types of tests verify the intended working order of components such as user interfaces, APIs, databases, and source code.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Adopt a test-driven development (TDD) approach to writing software, where you develop test cases to specify and validate your code. To start, create test cases for each function. If the test fails, you write new code to pass the test. This approach helps you validate the expected result of each function. Run unit tests and validate that they pass before you commit code to a source code repository.

Implement both unit and integration tests as part of the build, test, and deployment stages of the CI/CD pipeline. Automate testing, and automatically initiate tests whenever a new version of the application is ready to be deployed. If success criteria are not met, the pipeline is halted or rolled back.

If the application is a web or mobile app, perform automated integration testing on multiple desktop browsers or real devices. This approach is particularly useful to validate the compatibility and functionality of mobile apps across a diverse range of devices.

Implementation steps

1. Write unit tests before you write functional code (*test-driven development*, or TDD). Establish code guidelines so that writing and running unit tests are a non-functional coding requirement.
2. Create a suite of automated integration tests that cover the identified testable functionalities. These tests should simulate user interactions and validate the expected outcomes.
3. Create the necessary test environment to run the integration tests. This may include staging or pre-production environments that closely mimic the production environment.

4. Set up your source, build, test, and deploy stages using the AWS CodePipeline console or AWS Command Line Interface (CLI).
5. Deploy the application once the code has been built and tested. AWS CodeDeploy can deploy it to your staging (testing) and production environments. These environments may include Amazon EC2 instances, AWS Lambda functions, or on-premises servers. The same deployment mechanism should be used to deploy the application to all environments.
6. Monitor the progress of your pipeline and the status of each stage. Use quality checks to block the pipeline based on the status of your tests. You can also receive notifications for any pipeline stage failure or pipeline completion.
7. Continually monitor the results of the tests, and look for patterns, regressions or areas that require more attention. Use this information to improve the test suite, identify areas of the application that need more robust testing, and optimize the deployment process.

Resources

Related best practices:

- REL07-BP04 Load test your workload
- REL08-BP03 Integrate resiliency testing as part of your deployment
- REL12-BP04 Test resiliency using chaos engineering

Related documents:

- AWS Prescriptive Guidance: Test automation
- Continuous Delivery and Continuous Integration
- Indicators for functional testing
- Monitoring pipelines
- Use AWS CodePipeline with AWS CodeBuild to test code and run builds
- AWS Device Farm

REL08-BP03 Integrate resiliency testing as part of your deployment

Integrate resiliency testing by consciously introducing failures in your system to measure its capability in case of disruptive scenarios. Resilience tests are different from unit and function tests that are usually integrated in deployment cycles, as they focus on the identification of

unanticipated failures in your system. While it is safe to start with resiliency testing integration in pre-production, set a goal to implement these tests in production as a part of your *game days*.

Desired outcome: Resiliency testing helps build confidence in the system's ability to withstand degradation in production. Experiments identify weak points that could lead to failure, which helps you improve your system to automatically and efficiently mitigate failure and degradation.

Common anti-patterns:

- Lack of observability and monitoring in deployment processes
- Reliance on humans to resolve system failures
- Poor quality analysis mechanisms
- Focus on known issues in a system and a lack of experimentation to identify any unknowns
- Identification of failures, but no resolution
- No documentation of findings and runbooks

Benefits of establishing best practices: Resilience testing integrated in your deployments helps to identify unknown issues in the system that otherwise go unnoticed, which can lead to downtime in production. Identification of these unknowns in a system helps you document findings, integrate testing into your CI/CD process, and build runbooks, which simplify mitigation through efficient, repeatable mechanisms.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The most common resiliency testing forms that can be integrated in your system's deployments are disaster recovery and chaos engineering.

- Include updates to your disaster recovery plans and standard operating procedures (SOPs) with any significant deployment.
- Integrate reliability testing into your automated deployment pipelines. Services such asAWS Resilience Hubcan be integrated into your CI/CD pipeline to establish continuous resilience assessments that are automatically evaluated as part of every deployment.
- Define your applications in AWS Resilience Hub. Resilience assessments generate code snippets that help you create recovery procedures as AWS Systems Manager documents for your applications and provide a list of recommended Amazon CloudWatch monitors and alarms.

- Once your DR plans and SOPs are updated, complete disaster recovery testing to verify that they are effective. Disaster recovery testing helps you determine if you can restore your system after an event and return to normal operations. You can simulate various disaster recovery strategies and identify whether your planning is sufficient to meet your uptime requirements. Common disaster recovery strategies include backup and restore, pilot light, cold standby, warm standby, hot standby, and active-active, and they all differ in cost and complexity. Before disaster recovery testing, we recommend that you define your recovery time objective (RTO) and recovery point objective (RPO) to simplify the choice of strategy to simulate. AWS offers disaster recovery tools like AWS Elastic Disaster Recovery to help you get started with your planning and testing.
- Chaos engineering experiments introduce disruptions to the system, such as network outages and service failures. By simulating with controlled failures, you can discover your system's vulnerabilities while containing the impacts of the injected failures. Just like the other strategies, run controlled failure simulations in non-production environments using services like AWS Fault Injection Service to gain confidence before deploying in production.

Resources

Related documents:

- Experiment with failure using resilience testing to build recovery preparedness
- Continually assessing application resilience with AWS Resilience Hub and AWS CodePipeline
- Disaster recovery (DR) architecture on AWS, part 1: Strategies for recovery in the cloud
- Verify the resilience of your workloads using Chaos Engineering
- Principles of Chaos Engineering
- Chaos Engineering Workshop

Related videos:

- AWS re:Invent 2020: Testing Resilience using Chaos Engineering
- Improve Application Resilience with AWS Fault Injection Service
- Prepare & Protect Your Applications From Disruption With AWS Resilience Hub

REL08-BP04 Deploy using immutable infrastructure

Immutable infrastructure is a model that mandates that no updates, security patches, or configuration changes happen in-place on production workloads. When a change is needed, the architecture is built onto new infrastructure and deployed into production.

Follow an immutable infrastructure deployment strategy to increase the reliability, consistency, and reproducibility in your workload deployments.

Desired outcome: With immutable infrastructure, no in-place modifications are allowed to run infrastructure resources within a workload. Instead, when a change is needed, a new set of updated infrastructure resources containing all the necessary changes are deployed in parallel to your existing resources. This deployment is validated automatically, and if successful, traffic is gradually shifted to the new set of resources.

This deployment strategy applies to software updates, security patches, infrastructure changes, configuration updates, and application updates, among others.

Common anti-patterns:

- Implementing in-place changes to running infrastructure resources.

Benefits of establishing this best practice:

- **Increased consistency across environments:** Since there are no differences in infrastructure resources across environments, consistency is increased and testing is simplified.
- **Reduction in configuration drifts:** By replacing infrastructure resources with a known and version-controlled configuration, the infrastructure is set to a known, tested, and trusted state, avoiding configuration drifts.
- **Reliable atomic deployments:** Deployments either complete successfully or nothing changes, increasing consistency and reliability in the deployment process.
- **Simplified deployments:** Deployments are simplified because they don't need to support upgrades. Upgrades are just new deployments.
- **Safer deployments with fast rollback and recovery processes:** Deployments are safer because the previous working version is not changed. You can roll back to it if errors are detected.
- **Enhanced security posture:** By not allowing changes to infrastructure, remote access mechanisms (such as SSH) can be disabled. This reduces the attack vector, improving your organization's security posture.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Automation

When defining an immutable infrastructure deployment strategy, it is recommended to use automation as much as possible to increase reproducibility and minimize the potential of human error. For more detail, see REL08-BP05 Deploy changes with automation and Automating safe, hands-off deployments.

With infrastructure as code (IaC), infrastructure provisioning, orchestration, and deployment steps are defined in a programmatic, descriptive, and declarative way and stored in a source control system. Leveraging infrastructure as code makes it simpler to automate infrastructure deployment and helps achieve infrastructure immutability.

Deployment patterns

When a change in the workload is required, the immutable infrastructure deployment strategy mandates that a new set of infrastructure resources is deployed, including all necessary changes. It is important for this new set of resources to follow a rollout pattern that minimizes user impact. There are two main strategies for this deployment:

Canary deployment: The practice of directing a small number of your customers to the new version, usually running on a single service instance (the canary). You then deeply scrutinize any behavior changes or errors that are generated. You can remove traffic from the canary if you encounter critical problems and send the users back to the previous version. If the deployment is successful, you can continue to deploy at your desired velocity, while monitoring the changes for errors, until you are fully deployed. AWS CodeDeploy can be configured with a deployment configuration that allows a canary deployment.

Blue/green deployment: Similar to the canary deployment, except that a full fleet of the application is deployed in parallel. You alternate your deployments across the two stacks (blue and green). Once again, you can send traffic to the new version, and fall back to the old version if you see problems with the deployment. Commonly all traffic is switched at once, however you can also use fractions of your traffic to each version to dial up the adoption of the new version using the weighted DNS routing capabilities of Amazon Route 53. AWS CodeDeploy and AWS Elastic Beanstalk can be configured with a deployment configuration that allows a blue/green deployment.

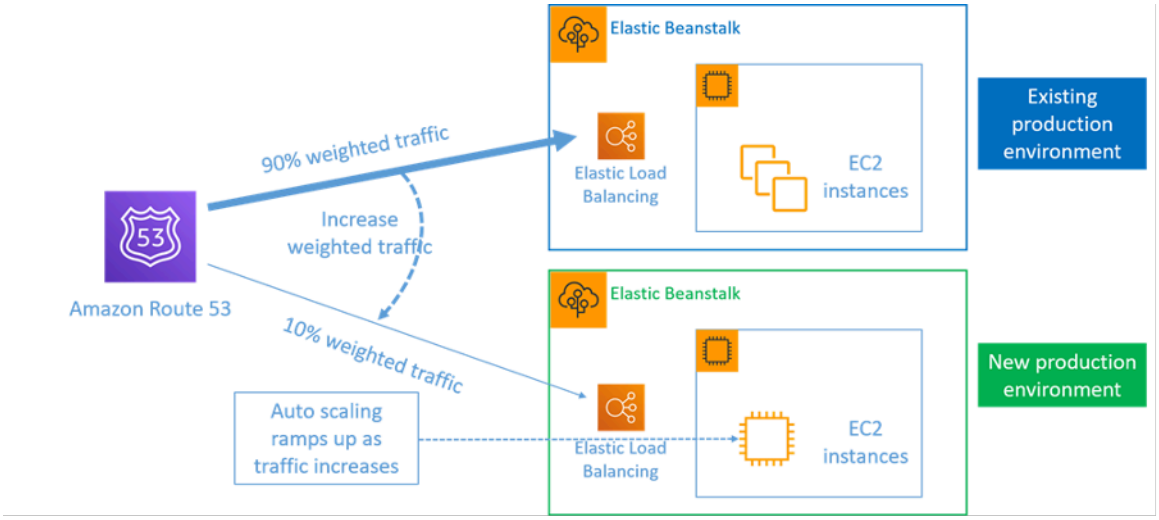


Figure 8: Blue/green deployment with AWS Elastic Beanstalk and Amazon Route 53

Drift detection

Drift is defined as any change that causes an infrastructure resource to have a different state or configuration to what is expected. Any type of unmanaged configuration change goes against the notion of immutable infrastructure, and should be detected and remediated in order to have a successful implementation of immutable infrastructure.

Implementation steps

- Disallow the in-place modification of running infrastructure resources.
  - You can use AWS Identity and Access Management (IAM) to specify who or what can access services and resources in AWS, centrally manage fine-grained permissions, and analyze access to refine permissions across AWS.
- Automate the deployment of infrastructure resources to increase reproducibility and minimize the potential of human error.
  - As described in the Introduction to DevOps on AWS whitepaper, automation is a cornerstone with AWS services and is internally supported in all services, features, and offerings.
  - *Prebaking* your Amazon Machine Image (AMI) can speed up the time to launch them. EC2 Image Builder is a fully managed AWS service that helps you automate the creation, maintenance, validation, sharing, and deployment of customized, secure, and up-to-date Linux or Windows custom AMI.
  - Some of the services that support automation are:

    - AWS Elastic Beanstalk is a service to rapidly deploy and scale web applications developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, NGINX, Passenger, and IIS.
    - AWS Proton helps platform teams connect and coordinate all the different tools your development teams need for infrastructure provisioning, code deployments, monitoring, and updates. AWS Proton enables automated infrastructure as code provisioning and deployment of serverless and container-based applications.
- Leveraging infrastructure as code makes it easy to automate infrastructure deployment, and helps achieve infrastructure immutability. AWS provides services that enable the creation, deployment, and maintenance of infrastructure in a programmatic, descriptive, and declarative way.
    - AWS CloudFormation helps developers create AWS resources in an orderly and predictable fashion. Resources are written in text files using JSON or YAML format. The templates require a specific syntax and structure that depends on the types of resources being created and managed. You author your resources in JSON or YAML with any code editor, check it into a version control system, and then CloudFormation builds the specified services in safe, repeatable manner.
    - AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS. AWS SAM integrates with other AWS services, and is an extension of CloudFormation.
    - AWS Cloud Development Kit (AWS CDK) is an open-source software development framework to model and provision your cloud application resources using familiar programming languages. You can use AWS CDK to model application infrastructure using TypeScript, Python, Java, and .NET. AWS CDK uses CloudFormation in the background to provision resources in a safe, repeatable manner.
    - AWS Cloud Control API introduces a common set of Create, Read, Update, Delete, and List (CRUDL) APIs to help developers manage their cloud infrastructure in an easy and consistent way. The Cloud Control API common APIs allow developers to uniformly manage the lifecycle of AWS and third-party services.
- Implement deployment patterns that minimize user impact.
    - Canary deployments:
        - Set up an API Gateway canary release deployment
        - Create a pipeline with canary deployments for Amazon ECS using AWS App Mesh

  - Blue/green deployments: the Blue/Green Deployments on AWS whitepaper describes example techniques to implement blue/green deployment strategies.
- Detect configuration or state drifts. For more detail, see Detecting unmanaged configuration changes to stacks and resources.

Resources

Related best practices:

- REL08-BP05 Deploy changes with automation

Related documents:

- Automating safe, hands-off deployments
- Leveraging AWS CloudFormation to create an immutable infrastructure at Nubank
- Infrastructure as code
- Implementing an alarm to automatically detect drift in AWS CloudFormation stacks

Related videos:

- AWS re:Invent 2020: Reliability, consistency, and confidence through immutability

REL08-BP05 Deploy changes with automation

Deployments and patching are automated to eliminate negative impact.

Making changes to production systems is one of the largest risk areas for many organizations. We consider deployments a first-class problem to be solved alongside the business problems that the software addresses. Today, this means the use of automation wherever practical in operations, including testing and deploying changes, adding or removing capacity, and migrating data.

Desired outcome: You build automated deployment safety into the release process with extensive pre-production testing, automatic rollbacks, and staggered production deployments. This automation minimizes the potential impact on production caused by failed deployments, and developers no longer need to actively watch deployments to production.

Common anti-patterns:

- You perform manual changes.
- You skip steps in your automation through manual emergency workflows.
- You don't follow your established plans and processes in favor of accelerated timelines.
- You perform rapid follow-on deployments without allowing for bake time.

Benefits of establishing this best practice: When you use automation to deploy all changes, you remove the potential for introduction of human error and provide the ability to test before you change production. Performing this process prior to production push verifies that your plans are complete. Additionally, automatic rollback into your release process can identify production issues and return your workload to its previously-working operational state.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Automate your deployment pipeline. Deployment pipelines allow you to invoke automated testing and detection of anomalies, and either halt the pipeline at a certain step before production deployment, or automatically roll back a change. An integral part of this is the adoption of the culture of continuous integration and continuous delivery/deployment (CI/CD), where a commit or code change passes through various automated stage gates from build and test stages to deployment on production environments.

Although conventional wisdom suggests that you keep people in the loop for the most difficult operational procedures, we suggest that you automate the most difficult procedures for that very reason.

Implementation steps

You can automate deployments to remove manual operations by following these steps:

- **Set up a code repository to store your code securely:** Use a hosted source code management system based on a popular technology such as Git to store your source code and infrastructure as code (IaC) configuration.
- **Configure a continuous integration service to compile your source code, run tests, and create deployment artifacts:** To set up a build project for this purpose, see Getting started with AWS CodeBuild using the console.
- **Set up a deployment service that automates application deployments and handles the complexity of application updates without reliance on error-prone manual deployments:**

AWS CodeDeploy automates software deployments to a variety of compute services, such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premise servers. To configure these steps, see Getting started with CodeDeploy.

- **Set up a continuous delivery service that automates your release pipelines for quicker and more reliable application and infrastructure updates:** Consider using AWS CodePipeline to help you automate your release pipelines. For more detail, see CodePipeline tutorials.

Resources

Related best practices:

- OPS05-BP04 Use build and deployment management systems
- OPS05-BP10 Fully automate integration and deployment
- OPS06-BP02 Test deployments
- OPS06-BP04 Automate testing and rollback

Related documents:

- Continuous Delivery of Nested AWS CloudFormation Stacks Using AWS CodePipeline
- APN Partner: partners that can help you create automated deployment solutions
- AWS Marketplace: products that can be used to automate your deployments
- Automate chat messages with webhooks.
- The Amazon Builders' Library: Ensuring rollback safety during deployments
- The Amazon Builders' Library: Going faster with continuous delivery
- What Is AWS CodePipeline?
- What Is CodeDeploy?
- AWS Systems Manager Patch Manager
- What is Amazon SES?
- What is Amazon Simple Notification Service?

Related videos:

- AWS Summit 2019: CI/CD on AWS

Failure management

Questions

- REL 9. How do you back up data?
- REL 10. How do you use fault isolation to protect your workload?
- REL 11. How do you design your workload to withstand component failures?
- REL 12. How do you test reliability?
- REL 13. How do you plan for disaster recovery (DR)?

REL 9. How do you back up data?

Back up data, applications, and configuration to meet your requirements for recovery time objectives (RTO) and recovery point objectives (RPO).

Best practices

- REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources
- REL09-BP02 Secure and encrypt backups
- REL09-BP03 Perform data backup automatically
- REL09-BP04 Perform periodic recovery of the data to verify backup integrity and processes

REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources

Understand and use the backup capabilities of the data services and resources used by the workload. Most services provide capabilities to back up workload data.

Desired outcome: Data sources have been identified and classified based on criticality. Then, establish a strategy for data recovery based on the RPO. This strategy involves either backing up these data sources, or having the ability to reproduce data from other sources. In the case of data loss, the strategy implemented allows recovery or the reproduction of data within the defined RPO and RTO.

Cloud maturity phase: Foundational

Common anti-patterns:

- Not aware of all data sources for the workload and their criticality.
- Not taking backups of critical data sources.
- Taking backups of only some data sources without using criticality as a criterion.
- No defined RPO, or backup frequency cannot meet RPO.
- Not evaluating if a backup is necessary or if data can be reproduced from other sources.

Benefits of establishing this best practice: Identifying the places where backups are necessary and implementing a mechanism to create backups, or being able to reproduce the data from an external source improves the ability to restore and recover data during an outage.

Level of risk exposed if this best practice is not established: High

Implementation guidance

All AWS data stores offer backup capabilities. Services such as Amazon RDS and Amazon DynamoDB additionally support automated backup that allows point-in-time recovery (PITR), which allows you to restore a backup to any time up to five minutes or less before the current time. Many AWS services offer the ability to copy backups to another AWS Region. AWS Backup is a tool that gives you the ability to centralize and automate data protection across AWS services. AWS Elastic Disaster Recovery allows you to copy full server workloads and maintain continuous data protection from on-premise, cross-AZ or cross-Region, with a Recovery Point Objective (RPO) measured in seconds.

Amazon S3 can be used as a backup destination for self-managed and AWS-managed data sources. AWS services such as Amazon EBS, Amazon RDS, and Amazon DynamoDB have built in capabilities to create backups. Third-party backup software can also be used.

On-premises data can be backed up to the AWS Cloud using AWS Storage Gateway or AWS DataSync. Amazon S3 buckets can be used to store this data on AWS. Amazon S3 offers multiple storage tiers such as Amazon Glacier or Amazon Glacier Deep Archive to reduce cost of data storage.

You might be able to meet data recovery needs by reproducing the data from other sources. For example, Amazon ElastiCache replica nodes or Amazon RDS read replicas could be used to reproduce data if the primary is lost. In cases where sources like this can be used to meet your Recovery Point Objective (RPO) and Recovery Time Objective (RTO), you might not require a backup. Another example, if working with Amazon EMR, it might not be necessary to backup your HDFS data store, as long as you can reproduce the data into Amazon EMR from Amazon S3.

When selecting a backup strategy, consider the time it takes to recover data. The time needed to recover data depends on the type of backup (in the case of a backup strategy), or the complexity of the data reproduction mechanism. This time should fall within the RTO for the workload.

Implementation steps

1. **Identify all data sources for the workload**. Data can be stored on a number of resources such as databases, volumes, filesystems, logging systems, and object storage. Refer to the **Resources** section to find **Related documents** on different AWS services where data is stored, and the backup capability these services provide.
2. **Classify data sources based on criticality**. Different data sets will have different levels of criticality for a workload, and therefore different requirements for resiliency. For example, some data might be critical and require a RPO near zero, while other data might be less critical and can tolerate a higher RPO and some data loss. Similarly, different data sets might have different RTO requirements as well.
3. **Use AWS or third-party services to create backups of the data**. AWS Backup is a managed service that allows creating backups of various data sources on AWS. AWS Elastic Disaster Recovery handles automated sub-second data replication to an AWS Region. Most AWS services also have native capabilities to create backups. The AWS Marketplace has many solutions that provide these capabilites as well. Refer to the **Resources** listed below for information on how to create backups of data from various AWS services.
4. **For data that is not backed up, establish a data reproduction mechanism**. You might choose not to backup data that can be reproduced from other sources for various reasons. There might be a situation where it is cheaper to reproduce data from sources when needed rather than creating a backup as there may be a cost associated with storing backups. Another example is where restoring from a backup takes longer than reproducing the data from sources, resulting in a breach in RTO. In such situations, consider tradeoffs and establish a well-defined process for how data can be reproduced from these sources when data recovery is necessary. For example, if you have loaded data from Amazon S3 to a data warehouse (like Amazon Redshift), or MapReduce cluster (like Amazon EMR) to do analysis on that data, this may be an example of data that can be reproduced from other sources. As long as the results of these analyses are either stored somewhere or reproducible, you would not suffer a data loss from a failure in the data warehouse or MapReduce cluster. Other examples that can be reproduced from sources include caches (like Amazon ElastiCache) or RDS read replicas.
5. **Establish a cadence for backing up data**. Creating backups of data sources is a periodic process and the frequency should depend on the RPO.

Level of effort for the Implementation Plan: Moderate

Resources

Related Best Practices:

REL13-BP01 Define recovery objectives for downtime and data loss

REL13-BP02 Use defined recovery strategies to meet the recovery objectives

Related documents:

- What Is AWS Backup?
- What is AWS DataSync?
- What is Volume Gateway?
- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Amazon EBS Snapshots
- Backing Up Amazon EFS
- Backing up Amazon FSx for Windows File Server
- Backup and Restore for ElastiCache for Redis
- Creating a DB Cluster Snapshot in Neptune
- Creating a DB Snapshot
- Creating an EventBridge Rule That Triggers on a Schedule
- Cross-Region Replication with Amazon S3
- EFS-to-EFS AWS Backup
- Exporting Log Data to Amazon S3
- Object lifecycle management
- On-Demand Backup and Restore for DynamoDB
- Point-in-time recovery for DynamoDB
- Working with Amazon OpenSearch Service Index Snapshots
- What is AWS Elastic Disaster Recovery?

Related videos:

- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS
- AWS Backup Demo: Cross-Account and Cross-Region Backup
- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP02 Secure and encrypt backups

Control and detect access to backups using authentication and authorization. Prevent and detect if data integrity of backups is compromised using encryption.

Implement security controls to prevent unauthorized access to backup data. Encrypt backups to protect the confidentiality and integrity of your data.

Common anti-patterns:

- Having the same access to the backups and restoration automation as you do to the data.
- Not encrypting your backups.
- Not implementing immutability for protection against deletion or tampering.
- Using the same security domain for production and backup systems.
- Not validating backup integrity through regular testing.

Benefits of establishing this best practice:

- Securing your backups prevents tampering with the data, and encryption of the data prevents access to that data if it is accidentally exposed.
- Enhanced protection against ransomware and other cyber threats that target backup infrastructure.
- Reduced recovery time following a cyber incident through validated recovery processes.
- Improved business continuity capabilities during security incidents.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Control and detect access to backups using authentication and authorization, such as AWS Identity and Access Management (IAM). Prevent and detect if data integrity of backups is compromised using encryption.

Amazon S3 supports several methods of encryption of your data at rest. Using server-side encryption, Amazon S3 accepts your objects as unencrypted data, and then encrypts them as they are stored. Using client-side encryption, your workload application is responsible for encrypting the data before it is sent to Amazon S3. Both methods allow you to use AWS Key Management Service (AWS KMS) to create and store the data key, or you can provide your own key, which you are then responsible for. Using AWS KMS, you can set policies using IAM on who can and cannot access your data keys and decrypted data.

For Amazon RDS, if you have chosen to encrypt your databases, then your backups are encrypted also. DynamoDB backups are always encrypted. When using AWS Elastic Disaster Recovery, all data in transit and at rest is encrypted. With Elastic Disaster Recovery, data at rest can be encrypted using either the default Amazon EBS encryption Volume Encryption Key or a custom customer-managed key.

Cyber resilience considerations

To enhance backup security against cyber threats, consider implementing these additional controls besides encryption:

- Implement immutability using AWS Backup Vault Lock or Amazon S3 Object Lock to prevent backup data from being altered or deleted during its retention period, protecting against ransomware and malicious deletion.
- Establish logical isolation between production and backup environments with AWS Backup logically air-gapped vault for critical systems, creating separation that helps prevent compromise of both environments simultaneously.
- Validate backup integrity regularly using AWS Backup restore testing to verify that backups are not corrupted and can be successfully restored following a cyber incident.
- Implement multi-party approval for critical recovery operations using AWS Backup multi-party approval to prevent unauthorized or malicious recovery attempts by requiring authorization from multiple designated approvers.

Implementation steps

1. Use encryption on each of your data stores. If your source data is encrypted, then the backup will also be encrypted.
    - Use encryption in Amazon RDS.. You can configure encryption at rest using AWS Key Management Service when you create an RDS instance.

- Use encryption on Amazon EBS volumes.. You can configure default encryption or specify a unique key upon volume creation.
- Use the required Amazon DynamoDB encryption. DynamoDB encrypts all data at rest. You can either use an AWS owned AWS KMS key or an AWS managed KMS key, specifying a key that is stored in your account.
- Encrypt your data stored in Amazon EFS. Configure the encryption when you create your file system.
- Configure the encryption in the source and destination Regions. You can configure encryption at rest in Amazon S3 using keys stored in KMS, but the keys are Region-specific. You can specify the destination keys when you configure the replication.
- Choose whether to use the default or custom Amazon EBS encryption for Elastic Disaster Recovery. This option will encrypt your replicated data at rest on the Staging Area Subnet disks and the replicated disks.

2. Implement least privilege permissions to access your backups. Follow best practices to limit the access to the backups, snapshots, and replicas in accordance with security best practices.
3. Configure immutability for critical backups. For critical data, implement AWS Backup Vault Lock or S3 Object Lock to prevent deletion or alteration during the specified retention period. For implementation details, see AWS Backup Vault Lock.
4. Create logical separation for backup environments. Implement AWS Backup logically air-gapped vault for critical systems requiring enhanced protection from cyber threats. For implementation guidance, see Building cyber resiliency with AWS Backup logically air-gapped vault.
5. Implement backup validation processes. Configure AWS Backup restore testing to regularly verify that backups are not corrupted and can be successfully restored following a cyber incident. For more information, see Validate recovery readiness with AWS Backup restore testing.
6. Configure multi-party approval for sensitive recovery operations. For critical systems, implement AWS Backup multi-party approval to require authorization from multiple designated approvers before recovery can proceed. For implementation details, see Improve recovery resilience with AWS Backup support for Multi-party approval.

Resources

Related documents:

- AWS Marketplace: products that can be used for backup
- Amazon EBS Encryption

- Amazon S3: Protecting Data Using Encryption
- CRR Additional Configuration: Replicating Objects Created with Server-Side Encryption (SSE) Using Encryption Keys stored in AWS KMS
- DynamoDB Encryption at Rest
- Encrypting Amazon RDS Resources
- Encrypting Data and Metadata in Amazon EFS
- Encryption for Backups in AWS
- Managing Encrypted Tables
- Security Pillar - AWS Well-Architected Framework
- What is AWS Elastic Disaster Recovery?
- FSISEC11: How are you protecting against ransomware?
- Ransomware Risk Management on AWS Using the NIST Cyber Security Framework
- Building cyber resiliency with AWS Backup logically air-gapped vault
- Validate recovery readiness with AWS Backup restore testing
- Improve recovery resilience with AWS Backup support for Multi-party approval

Related examples:

- Implementing Bi-Directional Cross-Region Replication (CRR) for Amazon S3

REL09-BP03 Perform data backup automatically

Configure backups to be taken automatically based on a periodic schedule informed by the Recovery Point Objective (RPO), or by changes in the dataset. Critical datasets with low data loss requirements need to be backed up automatically on a frequent basis, whereas less critical data where some loss is acceptable can be backed up less frequently.

Desired outcome: An automated process that creates backups of data sources at an established cadence.

Common anti-patterns:

- Performing backups manually.
- Using resources that have backup capability, but not including the backup in your automation.

Benefits of establishing this best practice: Automating backups verifies that they are taken regularly based on your RPO, and alerts you if they are not taken.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS Backup can be used to create automated data backups of various AWS data sources. Amazon RDS instances can be backed up almost continuously every five minutes and Amazon S3 objects can be backed up almost continuously every fifteen minutes, providing for point-in-time recovery (PITR) to a specific point in time within the backup history. For other AWS data sources, such as Amazon EBS volumes, Amazon DynamoDB tables, or Amazon FSx file systems, AWS Backup can run automated backup as frequently as every hour. These services also offer native backup capabilities. AWS services that offer automated backup with point-in-time recovery include Amazon DynamoDB, Amazon RDS, and Amazon Keyspaces (for Apache Cassandra) – these can be restored to a specific point in time within the backup history. Most other AWS data storage services offer the ability to schedule periodic backups, as frequently as every hour.

Amazon RDS and Amazon DynamoDB offer continuous backup with point-in-time recovery. Amazon S3 versioning, once turned on, is automatic. Amazon Data Lifecycle Manager can be used to automate the creation, copy and deletion of Amazon EBS snapshots. It can also automate the creation, copy, deprecation and deregistration of Amazon EBS-backed Amazon Machine Images (AMIs) and their underlying Amazon EBS snapshots.

AWS Elastic Disaster Recovery provides continuous block-level replication from the source environment (on-premises or AWS) to the target recovery region. Point-in-time Amazon EBS snapshots are automatically created and managed by the service.

For a centralized view of your backup automation and history, AWS Backup provides a fully managed, policy-based backup solution. It centralizes and automates the back up of data across multiple AWS services in the cloud as well as on premises using the AWS Storage Gateway.

In additional to versioning, Amazon S3 features replication. The entire S3 bucket can be automatically replicated to another bucket in the same, or a different AWS Region.

Implementation steps

1. **Identify data sources** that are currently being backed up manually. For more detail, see REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources.

2. **Determine the RPO** for the workload. For more detail, see REL13-BP01 Define recovery objectives for downtime and data loss.
3. **Use an automated backup solution or managed service**. AWS Backup is a fully-managed service that makes it easy to centralize and automate data protection across AWS services, in the cloud, and on-premises. Using backup plans in AWS Backup, create rules which define the resources to backup, and the frequency at which these backups should be created. This frequency should be informed by the RPO established in Step 2. For hands-on guidance on how to create automated backups using AWS Backup, see Testing Backup and Restore of Data. Native backup capabilities are offered by most AWS services that store data. For example, RDS can be leveraged for automated backups with point-in-time recovery (PITR).
4. **For data sources not supported** by an automated backup solution or managed service such as on-premises data sources or message queues, consider using a trusted third-party solution to create automated backups. Alternatively, you can create automation to do this using the AWS CLI or SDKs. You can use AWS Lambda Functions or AWS Step Functions to define the logic involved in creating a data backup, and use Amazon EventBridge to invoke it at a frequency based on your RPO.

Level of effort for the Implementation Plan: Low

Resources

Related documents:

- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Creating an EventBridge Rule That Triggers on a Schedule
- What Is AWS Backup?
- What Is AWS Step Functions?
- What is AWS Elastic Disaster Recovery?

Related videos:

- AWS re:Invent 2019: Deep dive on AWS Backup, ft. Rackspace (STG341)

REL09-BP04 Perform periodic recovery of the data to verify backup integrity and processes

Validate that your backup process implementation meets your Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) by performing a recovery test.

Desired outcome: Data from backups is periodically recovered using well-defined mechanisms to verify that recovery is possible within the established recovery time objective (RTO) for the workload. Verify that restoration from a backup results in a resource that contains the original data without any of it being corrupted or inaccessible, and with data loss within the recovery point objective (RPO).

Common anti-patterns:

- Restoring a backup, but not querying or retrieving any data to check that the restoration is usable.
- Assuming that a backup exists.
- Assuming that the backup of a system is fully operational and that data can be recovered from it.
- Assuming that the time to restore or recover data from a backup falls within the RTO for the workload.
- Assuming that the data contained on the backup falls within the RPO for the workload
- Restoring when necessary, without using a runbook or outside of an established automated procedure.

Benefits of establishing this best practice: Testing the recovery of the backups verifies that data can be restored when needed without having any worry that data might be missing or corrupted, that the restoration and recovery is possible within the RTO for the workload, and any data loss falls within the RPO for the workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Testing backup and restore capability increases confidence in the ability to perform these actions during an outage. Periodically restore backups to a new location and run tests to verify the integrity of the data. Some common tests that should be performed are checking if all data is available, is not corrupted, is accessible, and that any data loss falls within the RPO for the workload. Such tests can also help ascertain if recovery mechanisms are fast enough to accommodate the workload's RTO.

Using AWS, you can stand up a testing environment and restore your backups to assess RTO and RPO capabilities, and run tests on data content and integrity.

Additionally, Amazon RDS and Amazon DynamoDB allow point-in-time recovery (PITR). Using continuous backup, you can restore your dataset to the state it was in at a specified date and time.

If all the data is available, is not corrupted, is accessible, and any data loss falls within the RPO for the workload. Such tests can also help ascertain if recovery mechanisms are fast enough to accommodate the workload's RTO.

AWS Elastic Disaster Recovery offers continual point-in-time recovery snapshots of Amazon EBS volumes. As source servers are replicated, point-in-time states are chronicled over time based on the configured policy. Elastic Disaster Recovery helps you verify the integrity of these snapshots by launching instances for test and drill purposes without redirecting the traffic.

Implementation steps

1. **Identify data sources** that are currently being backed up and where these backups are being stored. For implementation guidance, see REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources.
2. **Establish criteria for data validation** for each data source. Different types of data will have different properties which might require different validation mechanisms. Consider how this data might be validated before you are confident to use it in production. Some common ways to validate data are using data and backup properties such as data type, format, checksum, size, or a combination of these with custom validation logic. For example, this might be a comparison of the checksum values between the restored resource and the data source at the time the backup was created.
3. **Establish RTO and RPO** for restoring the data based on data criticality. For implementation guidance, see REL13-BP01 Define recovery objectives for downtime and data loss.
4. **Assess your recovery capability**. Review your backup and restore strategy to understand if it can meet your RTO and RPO, and adjust the strategy as necessary. Using AWS Resilience Hub, you can run an assessment of your workload. The assessment evaluates your application configuration against the resiliency policy and reports if your RTO and RPO targets can be met.
5. **Do a test restore** using currently established processes used in production for data restoration. These processes depend on how the original data source was backed up, the format and storage location of the backup itself, or if the data is reproduced from other sources. For example, if you are using a managed service such as AWS Backup, this might be as simple as restoring the

backup into a new resource. If you used AWS Elastic Disaster Recovery you can launch a recovery drill.

6. **Validate data recovery** from the restored resource based on criteria you previously established for data validation. Does the restored and recovered data contain the most recent record or item at the time of backup? Does this data fall within the RPO for the workload?
7. **Measure time required** for restore and recovery and compare it to your established RTO. Does this process fall within the RTO for the workload? For example, compare the timestamps from when the restoration process started and when the recovery validation completed to calculate how long this process takes. All AWS API calls are timestamped and this information is available in AWS CloudTrail. While this information can provide details on when the restore process started, the end timestamp for when the validation was completed should be recorded by your validation logic. If using an automated process, then services like Amazon DynamoDB can be used to store this information. Additionally, many AWS services provide an event history which provides timestamped information when certain actions occurred. Within AWS Backup, backup and restore actions are referred to as *jobs*, and these jobs contain timestamp information as part of its metadata which can be used to measure time required for restoration and recovery.
8. **Notify stakeholders** if data validation fails, or if the time required for restoration and recovery exceeds the established RTO for the workload. When implementing automation to do this, such as in this lab, services like Amazon Simple Notification Service (Amazon SNS) can be used to send push notifications such as email or SMS to stakeholders. These messages can also be published to messaging applications such as Amazon Chime, Slack, or Microsoft Teams or used to create tasks as OpsItems using AWS Systems Manager OpsCenter.
9. **Automate this process to run periodically**. For example, services like AWS Lambda or a State Machine in AWS Step Functions can be used to automate the restore and recovery processes, and Amazon EventBridge can be used to invoke this automation workflow periodically as shown in the architecture diagram below. Learn how to Automate data recovery validation with AWS Backup. Additionally, this Well-Architected lab provides a hands-on experience on one way to do automation for several of the steps here.

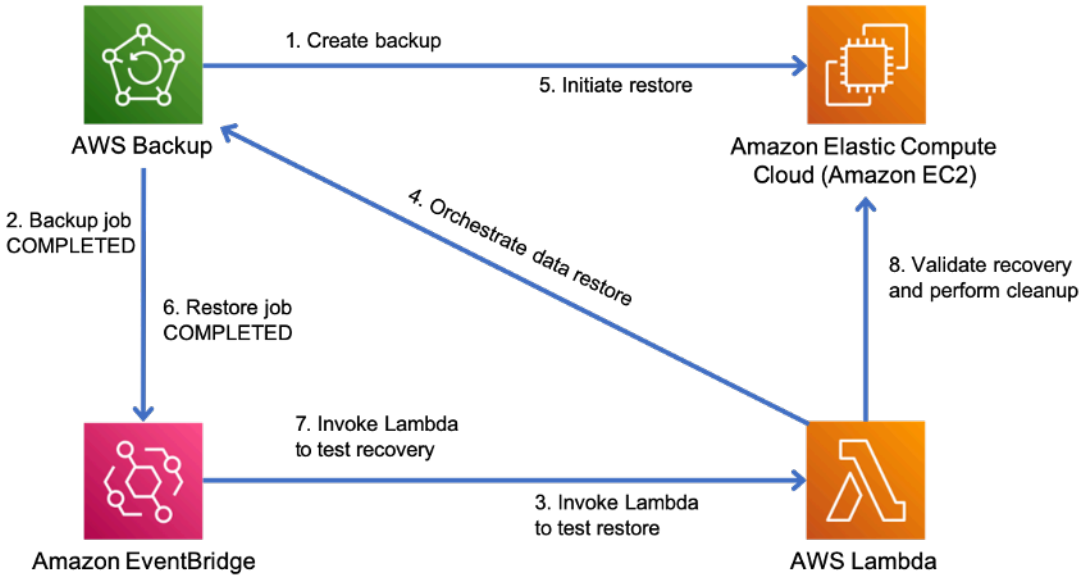


Figure 9. An automated backup and restore process

Level of effort for the Implementation Plan: Moderate to high depending on the complexity of the validation criteria.

Resources

Related documents:

- Automate data recovery validation with AWS Backup
- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Creating an EventBridge Rule That Triggers on a Schedule
- On-demand backup and restore for DynamoDB
- What Is AWS Backup?
- What Is AWS Step Functions?
- What is AWS Elastic Disaster Recovery
- AWS Elastic Disaster Recovery

REL 10. How do you use fault isolation to protect your workload?

Fault isolation limits the impact of a component or system failure to a defined boundary. With proper isolation, components outside of the boundary are unaffected by the failure. Running your workload across multiple fault isolation boundaries can make it more resilient to failure.

Best practices

- REL10-BP01 Deploy the workload to multiple locations
- REL10-BP02 Automate recovery for components constrained to a single location
- REL10-BP03 Use bulkhead architectures to limit scope of impact

REL10-BP01 Deploy the workload to multiple locations

Distribute workload data and resources across multiple Availability Zones or, where necessary, across AWS Regions.

A fundamental principle for service design in AWS is to avoid single points of failure, including the underlying physical infrastructure. AWS provides cloud computing resources and services globally across multiple geographic locations called Regions. Each Region is physically and logically independent and consists of three or more Availability Zones (AZs). Availability Zones are geographically close to each other but are physically separated and isolated. When you distribute your workloads among Availability Zones and Regions, you mitigate the risk of threats such as fires, floods, weather-related disasters, earthquakes, and human error.

Create a location strategy to provide high availability that is appropriate for your workloads.

Desired outcome: Production workloads are distributed among multiple Availability Zones (AZs) or Regions to achieve fault tolerance and high availability.

Common anti-patterns:

- Your production workload exists only in a single Availability Zone.
- You implement a multi-Region architecture when a multi-AZ architecture would satisfy business requirements.
- Your deployments or data become desynchronized, which results in configuration drift or under-replicated data.
- You don't account for dependencies between application components if resilience and multi-location requirements differ between those components.

Benefits of establishing this best practice:

- Your workload is more resilient to incidents, such as power or environmental control failures, natural disasters, upstream service failures, or network issues that impact an AZ or an entire Region.
- You can access a wider inventory of Amazon EC2 instances and reduce the likelihood of InsufficientCapacityExceptions (ICE) when launching specific EC2 instance types.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Deploy and operate all production workloads in at least two Availability Zones (AZs) in a Region.

Using multiple Availability Zones

Availability Zones are resource hosting locations that are physically separated from each other to avoid correlated failures due to risks such as fires, floods, and tornadoes. Each Availability Zone has independent physical infrastructure, including utility power connections, backup power sources, mechanical services, and network connectivity. This arrangement limits faults in any of these components to just the impacted Availability Zone. For example, if an AZ-wide incident makes EC2 instances unavailable in the affected Availability Zone, your instances in other Availability Zone remains available.

Despite being physically separated, Availability Zones in the same AWS Region are close enough to provide high-throughput, low-latency (single-digit millisecond) networking. You can replicate data synchronously between Availability Zones for most workloads without significantly impacting user experience. This means you can use Availability Zones in a Region in an active/active or active/standby configuration.

All compute associated with your workload should be distributed among multiple Availability Zones. This includes Amazon EC2 instances, AWS Fargate tasks, and VPC-attached AWS Lambda functions. AWS compute services, including EC2 Auto Scaling, Amazon Elastic Container Service (ECS), and Amazon Elastic Kubernetes Service (EKS), provide ways for you to launch and manage compute across Availability Zones. Configure them to automatically replace compute as needed in a different Availability Zone to maintain availability. To direct traffic to available Availability Zones, place a load balancer in front of your compute, such as an Application Load Balancer or Network Load Balancer. AWS load balancers can reroute traffic to available instances in the event of an Availability Zone impairment.

You should also replicate data for your workload and make it available in multiple Availability Zones. Some AWS managed data services, such as Amazon S3, Amazon Elastic File Service (EFS), Amazon Aurora, Amazon DynamoDB, Amazon Simple Queue Service (SQS), and Amazon Kinesis Data Streams replicate data in multiple Availability Zones by default and are robust against Availability Zone impairment. With other AWS managed data services, such as Amazon Relational Database Service (RDS), Amazon Redshift, and Amazon ElastiCache, you must enable multi-AZ replication. Once enabled, these services automatically detect an Availability Zone impairment, redirect requests to an available Availability Zone, and re-replicate data as needed after recovery without customer intervention. Familiarize yourself with the user guide for each AWS managed data service you use to understand its multi-AZ capabilities, behaviors, and operations.

If you are using self-managed storage, such as Amazon Elastic Block Store (EBS) volumes or Amazon EC2 instance storage, you must manage multi-AZ replication yourself.

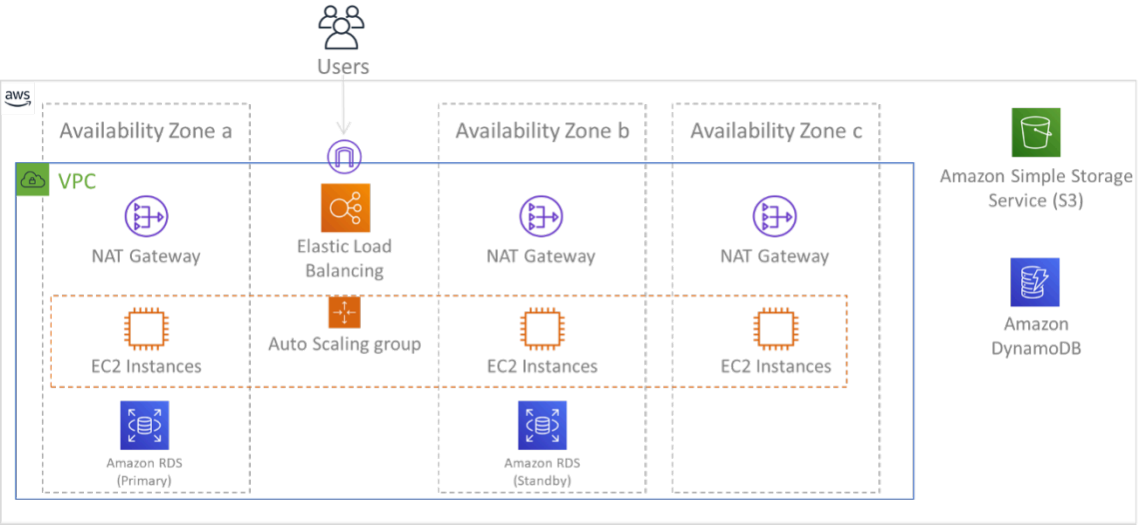


Figure 9: Multi-tier architecture deployed across three Availability Zones. Note that Amazon S3 and Amazon DynamoDB are always Multi-AZ automatically. The ELB also is deployed to all three zones.

Using multiple AWS Regions

If you have workloads that require extreme resilience (such as critical infrastructure, health-related applications, or services with stringent customer or mandated availability requirements), you may require additional availability beyond what a single AWS Region can provide. In this case, you should deploy and operate your workload across at least two AWS Regions (assuming that your data residency requirements allow it).

AWS Regions are located in different geographical regions around the world and in multiple continents. AWS Regions have even greater physical separation and isolation than Availability

Zones alone. AWS services, with few exceptions, take advantage of this design to operate fully independently between different Regions (also known as *Regional services*). A failure of an AWS Regional service is designed not to impact the service in a different Region.

When you operate your workload in multiple Regions, you should consider additional requirements. Because resources in different Regions are separate from and independent of one another, you must duplicate your workload's components in each Region. This includes foundational infrastructure, such as VPCs, in addition to compute and data services.

NOTE: When you consider a multi-Regional design, verify that your workload is capable of running in a single Region. If you create dependencies between Regions where a component in one Region relies on services or components in a different Region, you can increase the risk of failure and significantly weaken your reliability posture.

To ease multi-Regional deployments and maintain consistency, AWS CloudFormation StackSets can replicate your entire AWS infrastructure across multiple Regions. AWS CloudFormation can also detect configuration drift and inform you when your AWS resources in a Region are out of sync. Many AWS services offer multi-region replication for important workload assets. For example, EC2 Image Builder can publish your EC2 machine images (AMIs) after every build to each Region you use. Amazon Elastic Container Registry (ECR) can replicate your container images to your selected Regions.

You must also replicate your data across each of your chosen Regions. Many AWS managed data services provide cross-Regional replication capability, including Amazon S3, Amazon DynamoDB, Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon Elasticache, and Amazon EFS. Amazon DynamoDB global tables accept writes in any supported Region and will replicate data among all your other configured Regions. With other services, you must designate a primary Region for writes, as other Regions contain read-only replicas. For each AWS-managed data service your workload uses, refer to its user guide and developer guide to understand its multi-Region capabilities and limitations. Pay special attention to where writes must be directed, transactional capabilities and limitations, how replication is performed, and how to monitor synchronization between Regions.

AWS also provides the ability to route request traffic to your Regional deployments with great flexibility. For example, you can configure your DNS records using Amazon Route 53 to direct traffic to the closest available Region to the user. Alternatively, you can configure your DNS records in an active/standby configuration, where you designate one Region as primary and fall back to a Regional replica only if the primary Region becomes unhealthy. You can configure Route 53 health checks to detect unhealthy endpoints and perform automatic failover and additionally use Amazon

Application Recovery Controller (ARC) to provide a highly-available routing control for manually re-routing traffic as needed.

Even if you choose not to operate in multiple Regions for high availability, consider multiple Regions as part of your disaster recovery (DR) strategy. If possible, replicate your workload's infrastructure components and data in a *warm standby* or *pilot light* configuration in a secondary Region. In this design, you replicate baseline infrastructure from the primary Region such as VPCs, Auto Scaling groups, container orchestrators, and other components, but you configure the variable-sized components in the standby Region (such as the number of EC2 instances and database replicas) to be a minimally-operable size. You also arrange for continuous data replication from the primary Region to the standby Region. If an incident occurs, you can then scale out, or grow, the resources in the standby Region, and then promote it to become the primary Region.

Implementation steps

1. Work with business stakeholders and data residency experts to determine which AWS Regions can be used to host your resources and data.
2. Work with business and technical stakeholders to evaluate your workload, and determine whether its resilience needs can be met by a multi-AZ approach (single AWS Region) or if they require a multi-Region approach (if multiple Regions are permitted). The use of multiple Regions can achieve greater availability but can involve additional complexity and cost. Consider the following factors in your evaluation:
   a. **Business objectives and customer requirements**: How much downtime is permitted should a workload-impacting incident occur in an Availability Zone or a Region? Evaluate your recovery point objectives as discussed in REL13-BP01 Define recovery objectives for downtime and data loss.
   b. **Disaster recovery (DR) requirements**: What kind of potential disaster do you want to insure yourself against? Consider the possibility of data loss or long-term unavailability at different scopes of impact from a single Availability Zone to an entire Region. If you replicate data and resources across Availability Zones, and a single Availability Zone experiences a sustained failure, you can recover service in another Availability Zone. If you replicate data and resources across Regions, you can recover service in another Region.
3. Deploy your compute resources into multiple Availability Zones.
   a. In your VPC, create multiple subnets in different Availability Zones. Configure each to be large enough to accommodate the resources needed to serve the workload, even during an incident. For more detail, see REL02-BP03 Ensure IP subnet allocation accounts for expansion and availability.

   b. If you are using Amazon EC2 instances, use EC2 Auto Scaling to manage your instances. Specify the subnets you chose in the previous step when you create your Auto Scaling groups.

   c. If you are using AWS Fargate compute for Amazon ECS or Amazon EKS, select the subnets you chose in the first step when you create an ECS Service, launch an ECS task, or create a Fargate profile for EKS.

   d. If you are using AWS Lambda functions that need to run in your VPC, select the subnets you chose in the first step when you create the Lambda function. For any functions that do not have a VPC configuration, AWS Lambda manages availability for you automatically.

   e. Place traffic directors such as load balancers in front of your compute resources. If cross-zone load balancing is enabled, AWS Application Load Balancers and Network Load Balancers detect when targets such as EC2 instances and containers are unreachable due to Availability Zone impairment and reroute traffic towards targets in healthy Availability Zones. If you disable cross-zone load balancing, use Amazon Application Recovery Controller (ARC) to provide zonal shift capability. If you are using a third-party load balancer or have implemented your own load balancers, configure them with multiple front ends across different Availability Zones.

4. Replicate your workload's data across multiple Availability Zones.

   a. If you use an AWS-managed data service such as Amazon RDS, Amazon ElastiCache, or Amazon FSx, study its user guide to understand its data replication and resilience capabilities. Enable cross-AZ replication and failover if necessary.

   b. If you use AWS-managed storage services such as Amazon S3, Amazon EFS, and Amazon FSx, avoid using single-AZ or One Zone configurations for data that requires high durability. Use a multi-AZ configuration for these services. Check the respective service's user guide to determine whether multi-AZ replication is enabled by default or whether you must enable it.

   c. If you run a self-managed database, queue, or other storage service, arrange for multi-AZ replication according to the application's instructions or best practices. Familiarize yourself with the failover procedures for your application.

5. Configure your DNS service to detect AZ impairment and reroute traffic to a healthy Availability Zone. Amazon Route 53, when used in combination with Elastic Load Balancers, can do this automatically. Route 53 can also be configured with failover records that use health checks to respond to queries with only healthy IP addresses. For any DNS records used for failover, specify a short time to live (TTL) value (for example, 60 seconds or less) to help prevent record caching from impeding recovery (Route 53 alias records supply appropriate TTLs for you).

Additional steps when using multiple AWS Regions

1. Replicate all operating system (OS) and application code used by your workload across your selected Regions. Replicate Amazon Machine Images (AMIs) used by your EC2 instances if necessary using solutions such as Amazon EC2 Image Builder. Replicate container images stored in registries using solutions such as Amazon ECR cross-Region replication. Enable Regional replication for any Amazon S3 buckets used for storing application resources.
2. Deploy your compute resources and configuration metadata (such as parameters stored in AWS Systems Manager Parameter Store) into multiple Regions. Use the same procedures described in previous steps, but replicate the configuration for each Region you are using for your workload. Use infrastructure as code solutions such as AWS CloudFormation to uniformly reproduce the configurations among Regions. If you are using a secondary Region in a pilot light configuration for disaster recovery, you may reduce the number of your compute resources to a minimum value to save cost, with a corresponding increase in time to recovery.
3. Replicate your data from your primary Region into your secondary Regions.
    a. Amazon DynamoDB global tables provide global replicas of your data that can be written to from any supported Region. With other AWS-managed data services, such as Amazon RDS, Amazon Aurora, and Amazon Elasticache, you designate a primary (read/write) Region and replica (read-only) Regions. Consult the respective services' user and developer guides for details on Regional replication.
    b. If you are running a self-managed database, arrange for multi-Region replication according to the application's instructions or best practices. Familiarize yourself with the failover procedures for your application.
    c. If your workload uses AWS EventBridge, you may need to forward selected events from your primary Region to your secondary Regions. To do so, specify event buses in your secondary Regions as targets for matched events in your primary Region.
4. Consider whether and to what extent you want to use identical encryption keys across Regions. A typical approach that balances security and ease of use is to use Region-scoped keys for Region-local data and authentication, and use globally-scoped keys for encryption of data that is replicated among different Regions. AWS Key Management Service (KMS) supports multi-region keys to securely distribute and protect keys shared across Regions.
5. Consider AWS Global Accelerator to improve the availability of your application by directing traffic to Regions that contain healthy endpoints.

Resources

Related best practices:

- REL02-BP03 Ensure IP subnet allocation accounts for expansion and availability
- REL11-BP05 Use static stability to prevent bimodal behavior
- REL13-BP01 Define recovery objectives for downtime and data loss

Related documents:

- AWS Global Infrastructure
- White paper: AWS Fault Isolation Boundaries
- Resilience in Amazon EC2 Auto Scaling
- Amazon EC2 Auto Scaling: Example: Distribute instances across Availability Zones
- How EC2 Image Builder works
- How Amazon ECS places tasks on container instances (includes Fargate)
- Resilience in AWS Lambda
- Amazon S3: Replicating objects overview
- Private image replication in Amazon ECR
- Global Tables: Multi-Region Replication with DynamoDB
- Amazon Elasticache for Redis OSS: Replication across AWS Regions using global datastores
- Resilience in Amazon RDS
- Using Amazon Aurora global databases
- AWS Global Accelerator Developer Guide
- Multi-Region keys in AWS KMS
- Amazon Route 53: Configuring DNS failover
- Amazon Application Recovery Controller (ARC) Developer Guide
- Sending and receiving Amazon EventBridge events between AWS Regions
- Creating a Multi-Region Application with AWS Services blog series
- Disaster Recovery (DR) Architecture on AWS, Part I: Strategies for Recovery in the Cloud
- Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Innovation and operation of the AWS global network infrastructure

REL10-BP02 Automate recovery for components constrained to a single location

If components of the workload can only run in a single Availability Zone or in an on-premises data center, implement the capability to do a complete rebuild of the workload within your defined recovery objectives.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

If the best practice to deploy the workload to multiple locations is not possible due to technological constraints, you must implement an alternate path to resiliency. You must automate the ability to recreate necessary infrastructure, redeploy applications, and recreate necessary data for these cases.

For example, Amazon EMR launches all nodes for a given cluster in the same Availability Zone because running a cluster in the same zone improves performance of the jobs flows as it provides a higher data access rate. If this component is required for workload resilience, then you must have a way to redeploy the cluster and its data. Also for Amazon EMR, you should provision redundancy in ways other than using Multi-AZ. You can provision multiple nodes. Using EMR File System (EMRFS), data in EMR can be stored in Amazon S3, which in turn can be replicated across multiple Availability Zones or AWS Regions.

Similarly, for Amazon Redshift, by default it provisions your cluster in a randomly selected Availability Zone within the AWS Region that you select. All the cluster nodes are provisioned in the same zone.

For stateful server-based workloads deployed to an on-premise data center, you can use AWS Elastic Disaster Recovery to protect your workloads in AWS. If you are already hosted in AWS, you can use Elastic Disaster Recovery to protect your workload to an alternative Availability Zone or Region. Elastic Disaster Recovery uses continual block-level replication to a lightweight staging area to provide fast, reliable recovery of on-premises and cloud-based applications.

Implementation steps

1. Implement self-healing. Deploy your instances or containers using automatic scaling when possible. If you cannot use automatic scaling, use automatic recovery for EC2 instances or implement self-healing automation based on Amazon EC2 or ECS container lifecycle events.
    - Use Amazon EC2 Auto Scaling groups for instances and container workloads that have no requirements for a single instance IP address, private IP address, Elastic IP address, and instance metadata.
        - The launch template user data can be used to implement automation that can self-heal most workloads.
    - Use automatic recovery of Amazon EC2 instances for workloads that require a single instance ID address, private IP address, elastic IP address, and instance metadata.
        - Automatic Recovery will send recovery status alerts to a SNS topic as the instance failure is detected.
    - Use Amazon EC2 instance lifecycle events or Amazon ECS events to automate self-healing where automatic scaling or EC2 recovery cannot be used.
        - Use the events to invoke automation that will heal your component according to the process logic you require.
    - Protect stateful workloads that are limited to a single location using AWS Elastic Disaster Recovery.

Resources

Related documents:

- Amazon ECS events
- Amazon EC2 Auto Scaling lifecycle hooks
- Recover your instance.
- Service automatic scaling
- What Is Amazon EC2 Auto Scaling?
- AWS Elastic Disaster Recovery

REL10-BP03 Use bulkhead architectures to limit scope of impact

Implement bulkhead architectures (also known as cell-based architectures) to restrict the effect of failure within a workload to a limited number of components.

Desired outcome: A cell-based architecture uses multiple isolated instances of a workload, where each instance is known as a cell. Each cell is independent, does not share state with other cells, and handles a subset of the overall workload requests. This reduces the potential impact of a failure, such as a bad software update, to an individual cell and the requests it is processing. If a workload uses 10 cells to service 100 requests, when a failure occurs, 90% of the overall requests would be unaffected by the failure.

Common anti-patterns:

- Allowing cells to grow without bounds.
- Applying code updates or deployments to all cells at the same time.
- Sharing state or components between cells (with the exception of the router layer).
- Adding complex business or routing logic to the router layer.
- Not minimizing cross-cell interactions.

Benefits of establishing this best practice: With cell-based architectures, many common types of failure are contained within the cell itself, providing additional fault isolation. These fault boundaries can provide resilience against failure types that otherwise are hard to contain, such as unsuccessful code deployments or requests that are corrupted or invoke a specific failure mode (also known as *poison pill requests*).

Level of risk exposed if this best practice is not established: High

Implementation guidance

On a ship, bulkheads ensure that a hull breach is contained within one section of the hull. In complex systems, this pattern is often replicated to allow fault isolation. Fault isolated boundaries restrict the effect of a failure within a workload to a limited number of components. Components outside of the boundary are unaffected by the failure. Using multiple fault isolated boundaries, you can limit the impact on your workload. On AWS, customers can use multiple Availability Zones and Regions to provide fault isolation, but the concept of fault isolation can be extended to your workload's architecture as well.

The overall workload is partitioned cells by a partition key. This key needs to align with the *grain* of the service, or the natural way that a service's workload can be subdivided with minimal cross-cell interactions. Examples of partition keys are customer ID, resource ID, or any other parameter easily accessible in most API calls. A cell routing layer distributes requests to individual cells based on the partition key and presents a single endpoint to clients.

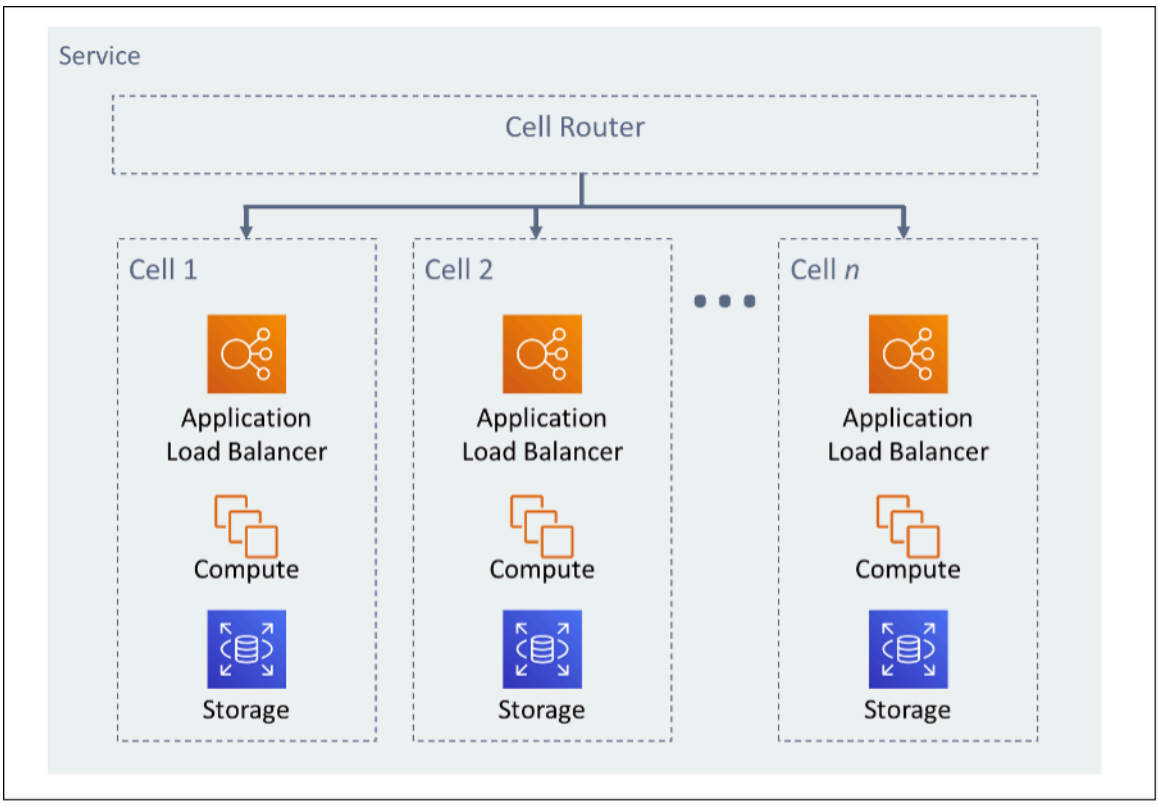


Figure 11: Cell-based architecture

Implementation steps

When designing a cell-based architecture, there are several design considerations to consider:

1. **Partition key**: Special consideration should be taken while choosing the partition key.
    - It should align with the grain of the service, or the natural way that a service's workload can be subdivided with minimal cross-cell interactions. Examples are `customer ID` or `resource ID`.
    - The partition key must be available in all requests, either directly or in a way that could be easily inferred deterministically by other parameters.
2. **Persistent cell mapping**: Upstream services should only interact with a single cell for the lifecycle of their resources.
    - Depending on the workload, a cell migration strategy may be needed to migrate data from one cell to another. A possible scenario when a cell migration may be needed is if a particular user or resource in your workload becomes too big and requires it to have a dedicated cell.
    - Cells should not share state or components between cells.

- Consequently, cross-cell interactions should be avoided or kept to a minimum, as those interactions create dependencies between cells and therefore diminish the fault isolation improvements.

3. **Router layer**: The router layer is a shared component between cells, and therefore cannot follow the same compartmentalization strategy as with cells.
   - It is recommended for the router layer to distribute requests to individual cells using a partition mapping algorithm in a computationally efficient manner, such as combining cryptographic hash functions and modular arithmetic to map partition keys to cells.
   - To avoid multi-cell impacts, the routing layer must remain as simple and horizontally scalable as possible, which necessitates avoiding complex business logic within this layer. This has the added benefit of making it easy to understand its expected behavior at all times, allowing for thorough testability. As explained by Colm MacCárthaigh in Reliability, constant work, and a good cup of coffee, simple designs and constant work patterns produce reliable systems and reduce anti-fragility.
4. **Cell size**: Cells should have a maximum size and should not be allowed to grow beyond it.
   - The maximum size should be identified by performing thorough testing, until breaking points are reached and safe operating margins are established. For more detail on how to implement testing practices, see REL07-BP04 Load test your workload
   - The overall workload should grow by adding additional cells, allowing the workload to scale with increases in demand.
5. **Multi-AZ or Multi-Region strategies**: Multiple layers of resilience should be leveraged to protect against different failure domains.
   - For resilience, you should use an approach that builds layers of defense. One layer protects against smaller, more common disruptions by building a highly available architecture using multiple AZs. Another layer of defense is meant to protect against rare events like widespread natural disasters and Region-level disruptions. This second layer involves architecting your application to span multiple AWS Regions. Implementing a multi-Region strategy for your workload helps protect it against widespread natural disasters that affect a large geographic region of a country, or technical failures of Region-wide scope. Be aware that implementing a multi-Region architecture can be significantly complex, and is usually not required for most workloads. For more detail, see REL10-BP01 Deploy the workload to multiple locations.
6. **Code deployment**: A staggered code deployment strategy should be preferred over deploying code changes to all cells at the same time.

- This helps minimize potential failure to multiple cells due to a bad deployment or human error. For more detail, see Automating safe, hands-off deployment.

Resources

Related best practices:

- REL07-BP04 Load test your workload
- REL10-BP01 Deploy the workload to multiple locations

Related documents:

- Reliability, constant work, and a good cup of coffee
- AWS and Compartmentalization
- Workload isolation using shuffle-sharding
- Automating safe, hands-off deployment

Related videos:

- AWS re:Invent 2018: Close Loops and Opening Minds: How to Take Control of Systems, Big and Small
- AWS re:Invent 2018: How AWS Minimizes the Blast Radius of Failures (ARC338)
- Shuffle-sharding: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)
- AWS Summit ANZ 2021 - Everything fails, all the time: Designing for resilience

REL 11. How do you design your workload to withstand component failures?

Workloads with a requirement for high availability and low mean time to recovery (MTTR) must be architected for resiliency.

Best practices

- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP02 Fail over to healthy resources
- REL11-BP03 Automate healing on all layers
- REL11-BP04 Rely on the data plane and not the control plane during recovery

- REL11-BP05 Use static stability to prevent bimodal behavior
- REL11-BP06 Send notifications when events impact availability
- REL11-BP07 Architect your product to meet availability targets and uptime service level agreements (SLAs)

REL11-BP01 Monitor all components of the workload to detect failures

Continually monitor the health of your workload so that you and your automated systems are aware of failures or degradations as soon as they occur. Monitor for key performance indicators (KPIs) based on business value.

All recovery and healing mechanisms must start with the ability to detect problems quickly. Technical failures should be detected first so that they can be resolved. However, availability is based on the ability of your workload to deliver business value, so key performance indicators (KPIs) that measure this need to be a part of your detection and remediation strategy.

Desired outcome: Essential components of a workload are monitored independently to detect and alert on failures when and where they happen.

Common anti-patterns:

- No alarms have been configured, so outages occur without notification.
- Alarms exist, but at thresholds that don't provide adequate time to react.
- Metrics are not collected often enough to meet the recovery time objective (RTO).
- Only the customer facing interfaces of the workload are actively monitored.
- Only collecting technical metrics, no business function metrics.
- No metrics measuring the user experience of the workload.
- Too many monitors are created.

Benefits of establishing this best practice: Having appropriate monitoring at all layers allows you to reduce recovery time by reducing time to detection.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify all workloads that will be reviewed for monitoring. Once you have identified all components of the workload that will need to monitored, you will now need to determine the

monitoring interval. The monitoring interval will have a direct impact on how fast recovery can be initiated based on the time it takes to detect a failure. The mean time to detection (MTTD) is the amount of time between a failure occurring and when repair operations begin. The list of services should be extensive and complete.

Monitoring must cover all layers of the application stack including application, platform, infrastructure, and network.

Your monitoring strategy should consider the impact of *gray failures*. For more detail on gray failures, see Gray failures in the Advanced Multi-AZ Resilience Patterns whitepaper.

Implementation steps

- Your monitoring interval is dependent on how quickly you must recover. Your recovery time is driven by the time it takes to recover, so you must determine the frequency of collection by accounting for this time and your recovery time objective (RTO).
- Configure detailed monitoring for components and managed services.
  - Determine if detailed monitoring for EC2 instances and Auto Scaling is necessary. Detailed monitoring provides one minute interval metrics, and default monitoring provides five minute interval metrics.
  - Determine if enhanced monitoring for RDS is necessary. Enhanced monitoring uses an agent on RDS instances to get useful information about different process or threads.
  - Determine the monitoring requirements of critical serverless components for Lambda, API Gateway, Amazon EKS, Amazon ECS, and all types of load balancers.
  - Determine the monitoring requirements of storage components for Amazon S3, Amazon FSx, Amazon EFS, and Amazon EBS.
- Create custom metrics to measure business key performance indicators (KPIs). Workloads implement key business functions, which should be used as KPIs that help identify when an indirect problem happens.
- Monitor the user experience for failures using user canaries. Synthetic transaction testing (also known as canary testing, but not to be confused with canary deployments) that can run and simulate customer behavior is among the most important testing processes. Run these tests constantly against your workload endpoints from diverse remote locations.
- Create custom metrics that track the user's experience. If you can instrument the experience of the customer, you can determine when the consumer experience degrades.

- Set alarms to detect when any part of your workload is not working properly and to indicate when to automatically scale resources. Alarms can be visually displayed on dashboards, send alerts through Amazon SNS or email, and work with Auto Scaling to scale workload resources up or down.
- Create dashboards to visualize your metrics. Dashboards can be used to visually see trends, outliers, and other indicators of potential problems or to provide an indication of problems you may want to investigate.
- Create distributed tracing monitoring for your services. With distributed monitoring, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors.
- Create monitoring systems (using CloudWatch or X-Ray) dashboards and data collection in a separate Region and account.
- Stay informed about service degradations with AWS Health. Create purpose-fit AWS Health event notifications to e-mail and chat channels through AWS User Notifications and integrate programmatically with your monitoring and alerting tools through Amazon EventBridge.

Resources

Related best practices:

- Availability Definition
- REL11-BP06 Send Notifications when events impact availability

Related documents:

- Amazon CloudWatch Synthetics enables you to create user canaries
- Enable or Disable Detailed Monitoring for Your Instance
- Enhanced Monitoring
- Monitoring Your Auto Scaling Groups and Instances Using Amazon CloudWatch
- Publishing Custom Metrics
- Using Amazon CloudWatch Alarms
- Using CloudWatch Dashboards
- Using Cross Region Cross Account CloudWatch Dashboards
- Using Cross Region Cross Account X-Ray Tracing

- Understanding availability

Related videos:

- Mitigating gray failures

Related examples:

- One Observability Workshop: Explore X-Ray

Related tools:

- CloudWatch
- CloudWatch X-Ray

REL11-BP02 Fail over to healthy resources

If a resource failure occurs, healthy resources should continue to serve requests. For location impairments (such as Availability Zone or AWS Region), ensure that you have systems in place to fail over to healthy resources in unimpaired locations.

When designing a service, distribute load across resources, Availability Zones, or Regions. Therefore, failure of an individual resource or impairment can be mitigated by shifting traffic to remaining healthy resources. Consider how services are discovered and routed to in the event of a failure.

Design your services with fault recovery in mind. At AWS, we design services to minimize the time to recover from failures and impact on data. Our services primarily use data stores that acknowledge requests only after they are durably stored across multiple replicas within a Region. They are constructed to use cell-based isolation and use the fault isolation provided by Availability Zones. We use automation extensively in our operational procedures. We also optimize our replace-and-restart functionality to recover quickly from interruptions.

The patterns and designs that allow for the failover vary for each AWS platform service. Many AWS native managed services are natively multiple Availability Zone (like Lambda or API Gateway). Other AWS services (like EC2 and EKS) require specific best practice designs to support failover of resources or data storage across AZs.

Monitoring should be set up to check that the failover resource is healthy, track the progress of the resources failing over, and monitor business process recovery.

Desired outcome: Systems are capable of automatically or manually using new resources to recover from degradation.

Common anti-patterns:

- Planning for failure is not part of the planning and design phase.
- RTO and RPO are not established.
- Insufficient monitoring to detect failing resources.
- Proper isolation of failure domains.
- Multi-Region fail over is not considered.
- Detection for failure is too sensitive or aggressive when deciding to failover.
- Not testing or validating failover design.
- Performing auto healing automation, but not notifying that healing was needed.
- Lack of dampening period to avoid failing back too soon.

Benefits of establishing this best practice: You can build more resilient systems that maintain reliability when experiencing failures by degrading gracefully and recovering quickly.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS services, such as Elastic Load Balancing and Amazon EC2 Auto Scaling, help distribute load across resources and Availability Zones. Therefore, failure of an individual resource (such as an EC2 instance) or impairment of an Availability Zone can be mitigated by shifting traffic to remaining healthy resources.

For multi-Region workloads, designs are more complicated. For example, cross-Region read replicas allow you to deploy your data to multiple AWS Regions. However, failover is still required to promote the read replica to primary and then point your traffic to the new endpoint. Amazon Route 53, Amazon Application Recovery Controller (ARC), Amazon CloudFront, and AWS Global Accelerator can help route traffic across AWS Regions.

AWS services, such as Amazon S3, Lambda, API Gateway, Amazon SQS, Amazon SNS, Amazon SES, Amazon Pinpoint, Amazon ECR, AWS Certificate Manager, EventBridge, or Amazon DynamoDB, are

automatically deployed to multiple Availability Zones by AWS. In case of failure, these AWS services automatically route traffic to healthy locations. Data is redundantly stored in multiple Availability Zones and remains available.

For Amazon RDS, Amazon Aurora, Amazon Redshift, Amazon EKS, or Amazon ECS, Multi-AZ is a configuration option. AWS can direct traffic to the healthy instance if failover is initiated. This failover action may be taken by AWS or as required by the customer

For Amazon EC2 instances, Amazon Redshift, Amazon ECS tasks, or Amazon EKS pods, you choose which Availability Zones to deploy to. For some designs, Elastic Load Balancing provides the solution to detect instances in unhealthy zones and route traffic to the healthy ones. Elastic Load Balancing can also route traffic to components in your on-premises data center.

For Multi-Region traffic failover, rerouting can leverage Amazon Route 53, Amazon Application Recovery Controller, AWS Global Accelerator, Route 53 Private DNS for VPCs, or CloudFront to provide a way to define internet domains and assign routing policies, including health checks, to route traffic to healthy Regions. AWS Global Accelerator provides static IP addresses that act as a fixed entry point to your application, then route to endpoints in AWS Regions of your choosing, using the AWS global network instead of the internet for better performance and reliability.

Implementation steps

- Create failover designs for all appropriate applications and services. Isolate each architecture component and create failover designs meeting RTO and RPO for each component.
- Configure lower environments (like development or test) with all services that are required to have a failover plan. Deploy the solutions using infrastructure as code (IaC) to ensure repeatability.
- Configure a recovery site such as a second Region to implement and test the failover designs. If necessary, resources for testing can be configured temporarily to limit additional costs.
- Determine which failover plans are automated by AWS, which can be automated by a DevOps process, and which might be manual. Document and measure each service's RTO and RPO.
- Create a failover playbook and include all steps to failover each resource, application, and service.
- Create a failback playbook and include all steps to failback (with timing) each resource, application, and service
- Create a plan to initiate and rehearse the playbook. Use simulations and chaos testing to test the playbook steps and automation.

- For location impairment (such as Availability Zone or AWS Region), ensure you have systems in place to fail over to healthy resources in unimpaired locations. Check quota, autoscaling levels, and resources running before failover testing.

Resources

Related Well-Architected best practices:

- REL13- Plan for DR
- REL10 - Use fault isolation to protect your workload

Related documents:

- Setting RTO and RPO Targets
- Failover using Route 53 Weighted routing
- Disaster Recovery with Amazon Application Recovery Controller
- EC2 with autoscaling
- EC2 Deployments - Multi-AZ
- ECS Deployments - Multi-AZ
- Switch traffic using Amazon Application Recovery Controller
- Lambda with an Application Load Balancer and Failover
- ACM Replication and Failover
- Parameter Store Replication and Failover
- ECR cross region replication and Failover
- Secrets manager cross region replication configuration
- Enable cross region replication for EFS and Failover
- EFS Cross Region Replication and Failover
- Networking Failover
- S3 Endpoint failover using MRAP
- Create cross region replication for S3
- Guidance for Cross Region Failover and Graceful Failback on AWS
- Failover using multi-region global accelerator

- Failover with DRS

Related examples:

- Disaster Recovery on AWS
- Elastic Disaster Recovery on AWS

REL11-BP03 Automate healing on all layers

Upon detection of a failure, use automated capabilities to perform actions to remediate. Degradations may be automatically healed through internal service mechanisms or require resources to be restarted or removed through remediation actions.

For self-managed applications and cross-Region healing, recovery designs and automated healing processes can be pulled from existing best practices.

The ability to restart or remove a resource is an important tool to remediate failures. A best practice is to make services stateless where possible. This prevents loss of data or availability on resource restart. In the cloud, you can (and generally should) replace the entire resource (for example, a compute instance or serverless function) as part of the restart. The restart itself is a simple and reliable way to recover from failure. Many different types of failures occur in workloads. Failures can occur in hardware, software, communications, and operations.

Restarting or retrying also applies to network requests. Apply the same recovery approach to both a network timeout and a dependency failure where the dependency returns an error. Both events have a similar effect on the system, so rather than attempting to make either event a special case, apply a similar strategy of limited retry with exponential backoff and jitter. Ability to restart is a recovery mechanism featured in recovery-oriented computing and high availability cluster architectures.

Desired outcome: Automated actions are performed to remediate detection of a failure.

Common anti-patterns:

- Provisioning resources without autoscaling.
- Deploying applications in instances or containers individually.
- Deploying applications that cannot be deployed into multiple locations without using automatic recovery.

- Manually healing applications that automatic scaling and automatic recovery fail to heal.
- No automation to failover databases.
- Lack automated methods to reroute traffic to new endpoints.
- No storage replication.

Benefits of establishing this best practice: Automated healing can reduce your mean time to recovery and improve your availability.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Designs for Amazon EKS or other Kubernetes services should include both minimum and maximum replica or stateful sets and the minimum cluster and node group sizing. These mechanisms provide a minimum amount of continually-available processing resources while automatically remediating any failures using the Kubernetes control plane.

Design patterns that are accessed through a load balancer using compute clusters should leverage Auto Scaling groups. Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones (AZs).

Clustered compute-based designs that do not use load balancing should have their size designed for loss of at least one node. This will allow for the service to maintain itself running in potentially reduced capacity while it's recovering a new node. Example services are Mongo, DynamoDB Accelerator, Amazon Redshift, Amazon EMR, Cassandra, Kafka, MSK-EC2, Couchbase, ELK, and Amazon OpenSearch Service. Many of these services can be designed with additional auto healing features. Some cluster technologies must generate an alert upon the loss a node triggering an automated or manual workflow to recreate a new node. This workflow can be automated using AWS Systems Manager to remediate issues quickly.

Amazon EventBridge can be used to monitor and filter for events such as CloudWatch alarms or changes in state in other AWS services. Based on event information, it can then invoke AWS Lambda, Systems Manager Automation, or other targets to run custom remediation logic on your workload. Amazon EC2 Auto Scaling can be configured to check for EC2 instance health. If the instance is in any state other than running, or if the system status is impaired, Amazon EC2 Auto Scaling considers the instance to be unhealthy and launches a replacement instance. For large-scale replacements (such as the loss of an entire Availability Zone), static stability is preferred for high availability.

Implementation steps

- Use Auto Scaling groups to deploy tiers in a workload. Auto Scaling can perform self-healing on stateless applications and add or remove capacity.
- For compute instances noted previously, use load balancing and choose the appropriate type of load balancer.
- Consider healing for Amazon RDS. With standby instances, configure for auto failover to the standby instance. For Amazon RDS Read Replica, automated workflow is required to make a read replica primary.
- Implement automatic recovery on EC2 instances that have applications deployed that cannot be deployed in multiple locations, and can tolerate rebooting upon failures. Automatic recovery can be used to replace failed hardware and restart the instance when the application is not capable of being deployed in multiple locations. The instance metadata and associated IP addresses are kept, as well as the EBS volumes and mount points to Amazon Elastic File System or File Systems for Lustre and Windows. Using AWS OpsWorks, you can configure automatic healing of EC2 instances at the layer level.
- Implement automated recovery using AWS Step Functions and AWS Lambda when you cannot use automatic scaling or automatic recovery, or when automatic recovery fails. When you cannot use automatic scaling, and either cannot use automatic recovery or automatic recovery fails, you can automate the healing using AWS Step Functions and AWS Lambda.
- Amazon EventBridge can be used to monitor and filter for events such as CloudWatch alarms or changes in state in other AWS services. Based on event information, it can then invoke AWS Lambda (or other targets) to run custom remediation logic on your workload.

Resources

Related best practices:

- Availability Definition
- REL11-BP01 Monitor all components of the workload to detect failures

Related documents:

- How AWS Auto Scaling Works
- Amazon EC2 Automatic Recovery
- Amazon Elastic Block Store (Amazon EBS)

- Amazon Elastic File System (Amazon EFS)
- What is Amazon FSx for Lustre?
- What is Amazon FSx for Windows File Server?
- AWS OpsWorks: Using Auto Healing to Replace Failed Instances
- What is AWS Step Functions?
- What is AWS Lambda?
- What Is Amazon EventBridge?
- Using Amazon CloudWatch Alarms
- Amazon RDS Failover
- SSM - Systems Manager Automation
- Resilient Architecture Best Practices

Related videos:

- Automatically Provision and Scale OpenSearch Service
- Amazon RDS Failover Automatically

Related examples:

- Amazon RDS Failover Workshop

Related tools:

- CloudWatch
- CloudWatch X-Ray

REL11-BP04 Rely on the data plane and not the control plane during recovery

Control planes provide the administrative APIs used to create, read and describe, update, delete, and list (CRUDL) resources, while data planes handle day-to-day service traffic. When implementing recovery or mitigation responses to potentially resiliency-impacting events, focus on using a minimal number of control plane operations to recover, rescale, restore, heal, or failover the service. Data plane action should supersede any activity during these degradation events.

For example, the following are all control plane actions: launching a new compute instance, creating block storage, and describing queue services. When you launch compute instances, the control plane has to perform multiple tasks like finding a physical host with capacity, allocating network interfaces, preparing local block storage volumes, generating credentials, and adding security rules. Control planes tend to be complicated orchestration.

Desired outcome: When a resource enters an impaired state, the system is capable of automatically or manually recovering by shifting traffic from impaired to healthy resources.

Common anti-patterns:

- Dependence on changing DNS records to re-route traffic.
- Dependence on control-plane scaling operations to replace impaired components due to insufficiently provisioned resources.
- Relying on extensive, multi service, multi-API control plane actions to remediate any category of impairment.

Benefits of establishing this best practice: Increased success rate for automated remediation can reduce your mean time to recovery and improve availability of the workload.

Level of risk exposed if this best practice is not established: Medium: For certain types of service degradations, control planes are affected. Dependencies on extensive use of the control plane for remediation may increase recovery time (RTO) and mean time to recovery (MTTR).

Implementation guidance

To limit data plane actions, assess each service for what actions are required to restore service.

Leverage Amazon Application Recovery Controller to shift the DNS traffic. These features continually monitor your application's ability to recover from failures and allow you to control your application recovery across multiple AWS Regions, Availability Zones, and on premises.

Route 53 routing policies use the control plane, so do not rely on it for recovery. The Route 53 data planes answer DNS queries and perform and evaluate health checks. They are globally distributed and designed for a 100% availability service level agreement (SLA).

The Route 53 management APIs and consoles where you create, update, and delete Route 53 resources run on control planes that are designed to prioritize the strong consistency and durability that you need when managing DNS. To achieve this, the control planes are located in a single Region: US East (N. Virginia). While both systems are built to be very reliable, the control planes

are not included in the SLA. There could be rare events in which the data plane's resilient design allows it to maintain availability while the control planes do not. For disaster recovery and failover mechanisms, use data plane functions to provide the best possible reliability.

Design your compute infrastructure to be statically stable to avoid using the control plane during an incident. For example, if you are using Amazon EC2 instances, avoid provisioning new instances manually or instructing Auto Scaling Groups to add instances in response. For the highest levels of resilience, provision sufficient capacity in the cluster used for failover. If this capacity threshold must be limited, set throttles on the overall end-to-end system to safely limit the total traffic reaching the limited set of resources.

For services like Amazon DynamoDB, Amazon API Gateway, load balancers, and AWS Lambda serverless, using those services leverages the data plane. However, creating new functions, load balancers, API gateways, or DynamoDB tables is a control plane action and should be completed before the degradation as preparation for an event and rehearsal of failover actions. For Amazon RDS, data plane actions allow for access to data.

For more information about data planes, control planes, and how AWS builds services to meet high availability targets, see Static stability using Availability Zones.

Understand which operations are on the data plane and which are on the control plane.

Implementation steps

For each workload that needs to be restored after a degradation event, evaluate the failover runbook, high availability design, auto healing design, or HA resource restoration plan. Identity each action that might be considered a control plane action.

Consider changing the control action to a data plane action:

- Auto Scaling (control plane) to pre-scaled Amazon EC2 resources (data plane)
- Amazon EC2 instance scaling (control plane) to AWS Lambda scaling (data plane)
- Assess any designs using Kubernetes and the nature of the control plane actions. Adding pods is a data plane action in Kubernetes. Actions should be limited to adding pods and not adding nodes. Using over-provisioned nodes is the preferred method to limit control plane actions

Consider alternate approaches that allow for data plane actions to affect the same remediation.

- Route 53 Record change (control plane) or Amazon Application Recovery Controller (data plane)

- Route 53 Health checks for more automated updates

Consider some services in a secondary Region, if the service is mission critical, to allow for more control plane and data plane actions in an unaffected Region.

- Amazon EC2 Auto Scaling or Amazon EKS in a primary Region compared to Amazon EC2 Auto Scaling or Amazon EKS in a secondary Region and routing traffic to secondary Region (control plane action)
- Make read replica in secondary primary or attempting same action in primary Region (control plane action)

Resources

Related best practices:

- Availability Definition
- REL11-BP01 Monitor all components of the workload to detect failures

Related documents:

- APN Partner: partners that can help with automation of your fault tolerance
- AWS Marketplace: products that can be used for fault tolerance
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Amazon DynamoDB API (control plane and data plane)
- AWS Lambda Executions (split into the control plane and the data plane)
- AWS Elemental MediaStore Data Plane
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack
- Creating Disaster Recovery Mechanisms Using Amazon Route 53
- What is Amazon Application Recovery Controller
- Kubernetes Control Plane and data plane

Related videos:

- Back to Basics - Using Static Stability
- Building resilient multi-site workloads using AWS global services

Related examples:

- Introducing Amazon Application Recovery Controller
- Amazon Builders' Library: Avoiding overload in distributed systems by putting the smaller service in control
- Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack
- Building highly resilient applications using Amazon Application Recovery Controller, Part 2: Multi-Region stack
- Static stability using Availability Zones

Related tools:

- Amazon CloudWatch
- AWS X-Ray

REL11-BP05 Use static stability to prevent bimodal behavior

Workloads should be statically stable and only operate in a single normal mode. Bimodal behavior is when your workload exhibits different behavior under normal and failure modes.

For example, you might try and recover from an Availability Zone failure by launching new instances in a different Availability Zone. This can result in a bimodal response during a failure mode. You should instead build workloads that are statically stable and operate within only one mode. In this example, those instances should have been provisioned in the second Availability Zone before the failure. This static stability design verifies that the workload only operates in a single mode.

Desired outcome: Workloads do not exhibit bimodal behavior during normal and failure modes.

Common anti-patterns:

- Assuming resources can always be provisioned regardless of the failure scope.
- Trying to dynamically acquire resources during a failure.
- Not provisioning adequate resources across zones or Regions until a failure occurs.
- Considering static stable designs for compute resources only.

Benefits of establishing this best practice: Workloads running with statically stable designs are capable of having predictable outcomes during normal and failure events.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Bimodal behavior occurs when your workload exhibits different behavior under normal and failure modes (for example, relying on launching new instances if an Availability Zone fails). An example of bimodal behavior is when stable Amazon EC2 designs provision enough instances in each Availability Zone to handle the workload load if one AZ were removed. Elastic Load Balancing or Amazon Route 53 health would check to shift a load away from the impaired instances. After traffic has shifted, use AWS Auto Scaling to asynchronously replace instances from the failed zone and launch them in the healthy zones. Static stability for compute deployment (such as EC2 instances or containers) results in the highest reliability.



Static stability of EC2 instances across Availability Zones

This must be weighed against the cost for this model and the business value of maintaining the workload under all resilience cases. It's less expensive to provision less compute capacity and rely on launching new instances in the case of a failure, but for large-scale failures (such as an

Availability Zone or Regional impairment), this approach is less effective because it relies on both an operational plane, and sufficient resources being available in the unaffected zones or Regions.

Your solution should also weigh reliability against the costs needs for your workload. Static stability architectures apply to a variety of architectures including compute instances spread across Availability Zones, database read replica designs, Kubernetes (Amazon EKS) cluster designs, and multi-Region failover architectures.

It is also possible to implement a more statically stable design by using more resources in each zone. By adding more zones, you reduce the amount of additional compute you need for static stability.

An example of bimodal behavior would be a network timeout that could cause a system to attempt to refresh the configuration state of the entire system. This would add an unexpected load to another component and might cause it to fail, resulting in other unexpected consequences. This negative feedback loop impacts the availability of your workload. Instead, you can build systems that are statically stable and operate in only one mode. A statically stable design would do constant work and always refresh the configuration state on a fixed cadence. When a call fails, the workload would use the previously cached value and initiate an alarm.

Another example of bimodal behavior is allowing clients to bypass your workload cache when failures occur. This might seem to be a solution that accommodates client needs but it can significantly change the demands on your workload and is likely to result in failures.

Assess critical workloads to determine what workloads require this type of resilience design. For those that are deemed critical, each application component must be reviewed. Example types of services that require static stability evaluations are:

- **Compute**: Amazon EC2, EKS-EC2, ECS-EC2, EMR-EC2
- **Databases**: Amazon Redshift, Amazon RDS, Amazon Aurora
- **Storage**: Amazon S3 (Single Zone), Amazon EFS (mounts), Amazon FSx (mounts)
- **Load balancers:** Under certain designs

Implementation steps

- Build systems that are statically stable and operate in only one mode. In this case, provision enough instances in each Availability Zone or Region to handle the workload capacity if one Availability Zone or Region were removed. A variety of services can be used for routing to healthy resources, such as:

  - Cross Region DNS Routing
  - MRAP Amazon S3 MultiRegion Routing
  - AWS Global Accelerator
  - Amazon Application Recovery Controller
- Configure database read replicas to account for the loss of a single primary instance or a read replica. If traffic is being served by read replicas, the quantity in each Availability Zone and each Region should equate to the overall need in case of the zone or Region failure.
- Configure critical data in Amazon S3 storage that is designed to be statically stable for data stored in case of an Availability Zone failure. If Amazon S3 One Zone-IA storage class is used, this should not be considered statically stable, as the loss of that zone minimizes access to this stored data.
- Load balancers are sometimes configured incorrectly or by design to service a specific Availability Zone. In this case, the statically stable design might be to spread a workload across multiple AZs in a more complex design. The original design may be used to reduce interzone traffic for security, latency, or cost reasons.

Resources

Related Well-Architected best practices:

- Availability Definition
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP04 Rely on the data plane and not the control plane during recovery

Related documents:

- Minimizing Dependencies in a Disaster Recovery Plan
- The Amazon Builders' Library: Static stability using Availability Zones
- Fault Isolation Boundaries
- Static stability using Availability Zones
- Multi-Zone RDS
- Minimizing Dependencies in a Disaster Recovery Plan
- Cross Region DNS Routing
- MRAP Amazon S3 MultiRegion Routing

- AWS Global Accelerator
- Amazon Application Recovery Controller
- Single Zone Amazon S3
- Cross Zone Load Balancing

Related videos:

- Static stability in AWS: AWS re:Invent 2019: Introducing The Amazon Builders' Library (DOP328)

REL11-BP06 Send notifications when events impact availability

Notifications are sent upon the detection of thresholds breached, even if the event causing the issue was automatically resolved.

Automated healing allows your workload to be reliable. However, it can also obscure underlying problems that need to be addressed. Implement appropriate monitoring and events so that you can detect patterns of problems, including those addressed by auto healing, so that you can resolve root cause issues.

Resilient systems are designed so that degradation events are immediately communicated to the appropriate teams. These notifications should be sent through one or many communication channels.

Desired outcome: Alerts are immediately sent to operations teams when thresholds are breached, such as error rates, latency, or other critical key performance indicator (KPI) metrics, so that these issues are resolved as soon as possible and user impact is avoided or minimized.

Common anti-patterns:

- Sending too many alarms.
- Sending alarms that are not actionable.
- Setting alarm thresholds too high (over sensitive) or too low (under sensitive).
- Not sending alarms for external dependencies.
- Not considering gray failures when designing monitoring and alarms.
- Performing healing automation, but not notifying the appropriate team that healing was needed.

Benefits of establishing this best practice: Notifications of recovery make operational and business teams aware of service degradations so that they can react immediately to minimize both mean time to detect (MTTD) and mean time to repair (MTTR). Notifications of recovery events also assure that you don't ignore problems that occur infrequently.

Level of risk exposed if this best practice is not established: Medium. Failure to implement appropriate monitoring and events notification mechanisms can result in failure to detect patterns of problems, including those addressed by auto healing. A team will only be made aware of system degradation when users contact customer service or by chance.

Implementation guidance

When defining a monitoring strategy, a triggered alarm is a common event. This event would likely contain an identifier for the alarm, the alarm state (such as `IN ALARM` or `OK`), and details of what triggered it. In many cases, an alarm event should be detected and an email notification sent. This is an example of an action on an alarm. Alarm notification is critical in observability, as it informs the right people that there is an issue. However, when action on events mature in your observability solution, it can automatically remediate the issue without the need for human intervention.

Once KPI-monitoring alarms have been established, alerts should be sent to appropriate teams when thresholds are exceeded. Those alerts may also be used to trigger automated processes that will attempt to remediate the degradation.

For more complex threshold monitoring, composite alarms should be considered. Composite alarms use a number of KPI-monitoring alarms to create an alert based on operational business logic. CloudWatch Alarms can be configured to send emails, or to log incidents in third-party incident tracking systems using Amazon SNS integration or Amazon EventBridge.

Implementation steps

Create various types of alarms based on how the workloads are monitored, such as:

- Application alarms are used to detect when any part of your workload is not working properly.
- Infrastructure alarms indicate when to scale resources. Alarms can be visually displayed on dashboards, send alerts through Amazon SNS or email, and work with Auto Scaling to scale workload resources in or out.
- Simple static alarms can be created to monitor when a metric breaches a static threshold for a specified number of evaluation periods.

- Composite alarms can account for complex alarms from multiple sources.
- Once the alarm has been created, create appropriate notification events. You can directly invoke an Amazon SNS API to send notifications and link any automation for remediation or communication.
- Stay informed about service degradations with AWS Health. Create purpose-fit AWS Health event notifications to e-mail and chat channels through AWS User Notifications and integrate programmatically with your monitoring and alerting tools through Amazon EventBridge.

Resources

Related Well-Architected best practices:

- Availability Definition

Related documents:

- Creating a CloudWatch Alarm Based on a Static Threshold
- What Is Amazon EventBridge?
- What is Amazon Simple Notification Service?
- Publishing Custom Metrics
- Using Amazon CloudWatch Alarms
- Setup CloudWatch Composite alarms
- What's new in AWS Observability at re:Invent 2022

Related tools:

- CloudWatch
- CloudWatch X-Ray

REL11-BP07 Architect your product to meet availability targets and uptime service level agreements (SLAs)

Architect your product to meet availability targets and uptime service level agreements (SLAs). If you publish or privately agree to availability targets or uptime SLAs, verify that your architecture and operational processes are designed to support them.

Desired outcome: Each application has a defined target for availability and SLA for performance metrics, which can be monitored and maintained in order to meet business outcomes.

Common anti-patterns:

- Designing and deploying workload's without setting any SLAs.
- SLA metrics are set too high without rationale or business requirements.
- Setting SLAs without taking into account for dependencies and their underlying SLA.
- Application designs are created without considering the Shared Responsibility Model for Resilience.

Benefits of establishing this best practice: Designing applications based on key resiliency targets helps you meet business objectives and customer expectations. These objectives help drive the application design process that evaluates different technologies and considers various tradeoffs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Application designs have to account for a diverse set of requirements that are derived from business, operational, and financial objectives. Within the operational requirements, workloads need to have specific resilience metric targets so they can be properly monitored and supported. Resilience metrics should not be set or derived after deploying the workload. They should be defined during the design phase and help guide various decisions and tradeoffs.

- Every workload should have its own set of resilience metrics. Those metrics may be different from other business applications.
- Reducing dependencies can have a positive impact on availability. Each workload should consider its dependencies and their SLAs. In general, select dependencies with availability goals equal to or greater than the goals of your workload.
- Consider loosely coupled designs so your workload can operate correctly despite dependency impairment, where possible.
- Reduce control plane dependencies, especially during recovery or a degradation. Evaluate designs that are statically stable for mission critical workloads. Use resource sparing to increase the availability of those dependencies in a workload.
- Observability and instrumentation are critical for achieving SLAs by reducing Mean Time to Detection (MTTD) and Mean Time to Repair (MTTR).

- Less frequent failure (longer MTBF), shorter failure detection times (shorter MTTD), and shorter repair times (shorter MTTR) are the three factors that are used to improve availability in distributed systems.
- Establishing and meeting resilience metrics for a workload is foundational to any effective design. Those designs must factor in tradeoffs of design complexity, service dependencies, performance, scaling, and costs.

Implementation steps

- Review and document the workload design considering the following questions:
  - Where are control planes used in the workload?
  - How does the workload implement fault tolerance?
  - What are the design patterns for scaling, automatic scaling, redundancy, and highly available components?
  - What are the requirements for data consistency and availability?
  - Are there considerations for resource sparing or resource static stability?
  - What are the service dependencies?
- Define SLA metrics based on the workload architecture while working with stakeholders. Consider the SLAs of all dependencies used by the workload.
- Once the SLA target has been set, optimize the architecture to meet the SLA.
- Once the design is set that will meet the SLA, implement operational changes, process automation, and runbooks that also will have focus on reducing MTTD and MTTR.
- Once deployed, monitor and report on the SLA.

Resources

Related best practices:

- REL03-BP01 Choose how to segment your workload
- REL10-BP01 Deploy the workload to multiple locations
- REL11-BP01 Monitor all components of the workload to detect failures
- REL11-BP03 Automate healing on all layers
- REL12-BP04 Test resiliency using chaos engineering
- REL13-BP01 Define recovery objectives for downtime and data loss

- Understanding workload health

Related documents:

- Availability with redundancy
- Reliability pillar - Availability
- Measuring availability
- AWS Fault Isolation Boundaries
- Shared Responsibility Model for Resiliency
- Static stability using Availability Zones
- AWS Service Level Agreements (SLAs)
- Guidance for Cell-based Architecture on AWS
- AWS infrastructure
- Advanced Multi-AZ Resilience Patterns whitepaper

Related services:

- Amazon CloudWatch
- AWS Config
- AWS Trusted Advisor

REL 12. How do you test reliability?

After you have designed your workload to be resilient to the stresses of production, testing is the only way to verify that it will operate as designed, and deliver the resiliency you expect.

Best practices

- REL12-BP01 Use playbooks to investigate failures
- REL12-BP02 Perform post-incident analysis
- REL12-BP03 Test scalability and performance requirements
- REL12-BP04 Test resiliency using chaos engineering
- REL12-BP05 Conduct game days regularly

REL12-BP01 Use playbooks to investigate failures

Permit consistent and prompt responses to failure scenarios that are not well understood, by documenting the investigation process in playbooks. Playbooks are the predefined steps performed to identify the factors contributing to a failure scenario. The results from any process step are used to determine the next steps to take until the issue is identified or escalated.

The playbook is proactive planning that you must do, to be able to take reactive actions effectively. When failure scenarios not covered by the playbook are encountered in production, first address the issue (put out the fire). Then go back and look at the steps you took to address the issue and use these to add a new entry in the playbook.

Note that playbooks are used in response to specific incidents, while runbooks are used to achieve specific outcomes. Often, runbooks are used for routine activities and playbooks are used to respond to non-routine events.

Common anti-patterns:

- Planning to deploy a workload without knowing the processes to diagnose issues or respond to incidents.
- Unplanned decisions about which systems to gather logs and metrics from when investigating an event.
- Not retaining metrics and events long enough to be able to retrieve the data.

Benefits of establishing this best practice: Capturing playbooks ensures that processes can be consistently followed. Codifying your playbooks limits the introduction of errors from manual activity. Automating playbooks shortens the time to respond to an event by eliminating the requirement for team member intervention or providing them additional information when their intervention begins.

Level of risk exposed if this best practice is not established: High

Implementation guidance

- Use playbooks to identify issues. Playbooks are documented processes to investigate issues. Allow consistent and prompt responses to failure scenarios by documenting processes in playbooks. Playbooks must contain the information and guidance necessary for an adequately skilled person to gather applicable information, identify potential sources of failure, isolate faults, and determine contributing factors (perform post-incident analysis).

- Implement playbooks as code. Perform your operations as code by scripting your playbooks to ensure consistency and limit reduce errors caused by manual processes. Playbooks can be composed of multiple scripts representing the different steps that might be necessary to identify the contributing factors to an issue. Runbook activities can be invoked or performed as part of playbook activities, or might prompt to run a playbook in response to identified events.
  - Automate your operational playbooks with AWS Systems Manager
  - AWS Systems Manager Run Command
  - AWS Systems Manager Automation
  - What is AWS Lambda?
  - What Is Amazon EventBridge?
  - Using Amazon CloudWatch Alarms

Resources

Related documents:

- AWS Systems Manager Automation
- AWS Systems Manager Run Command
- Automate your operational playbooks with AWS Systems Manager
- Using Amazon CloudWatch Alarms
- Using Canaries (Amazon CloudWatch Synthetics)
- What Is Amazon EventBridge?
- What is AWS Lambda?

Related examples:

- Automating operations with Playbooks and Runbooks

REL12-BP02 Perform post-incident analysis

Review customer-impacting events, and identify the contributing factors and preventative action items. Use this information to develop mitigations to limit or prevent recurrence. Develop procedures for prompt and effective responses. Communicate contributing factors and corrective

actions as appropriate, tailored to target audiences. Have a method to communicate these causes to others as needed.

Assess why existing testing did not find the issue. Add tests for this case if tests do not already exist.

Desired outcome: Your teams have a consistent and agreed upon approach to handling post-incident analysis. One mechanism is the correction of error (COE) process. The COE process helps your teams identify, understand, and address the root causes for incidents, while also building mechanisms and guardrails to limit the probability of the same incident happening again.

Common anti-patterns:

- Finding contributing factors, but not continuing to look deeper for other potential problems and approaches to mitigate.
- Only identifying human error causes, and not providing any training or automation that could prevent human errors.
- Focus on assigning blame rather than understanding the root cause, creating a culture of fear and hindering open communication
- Failure to share insights, which keeps incident analysis findings within a small group and prevents others from benefiting from the lessons learned
- No mechanism to capture institutional knowledge, thereby losing valuable insights by not preserving the lessons-learned in the form of updated best practices and resulting in repeat incidents with the same or similar root cause

Benefits of establishing this best practice: Conducting post-incident analysis and sharing the results permits other workloads to mitigate the risk if they have implemented the same contributing factors, and allows them to implement the mitigation or automated recovery before an incident occurs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Good post-incident analysis provides opportunities to propose common solutions for problems with architecture patterns that are used in other places in your systems.

A cornerstone of the COE process is documenting and addressing issues. It is recommended to define a standardized way to document critical root causes, and ensure they are reviewed and

addressed. Assign clear ownership for the post-incident analysis process. Designate a responsible team or individual who will oversee incident investigations and follow-ups.

Encourage a culture that focuses on learning and improvement rather than assigning blame. Emphasize that the goal is to prevent future incidents, not to penalize individuals.

Develop well-defined procedures for conducting post-incident analyses. These procedures should outline the steps to be taken, the information to be collected, and the key questions to be addressed during the analysis. Investigate incidents thoroughly, going beyond immediate causes to identify root causes and contributing factors. Use techniques like the *five whys* to delve deep into the underlying issues.

Maintain a repository of lessons learned from incident analyses. This institutional knowledge can serve as a reference for future incidents and prevention efforts. Share findings and insights from post-incident analyses, and consider holding open-invite post-incident review meetings to discuss lessons learned.

Implementation steps

- While conducting post-incident analysis, ensure the process is blame-free. This allows people involved in the incident to be dispassionate about the proposed corrective actions and promote honest self-assessment and collaboration across teams.
- Define a standardized way to document critical issues. An example structure for such document is as follows:
  - What happened?
  - What was the impact on customers and your business?
  - What was the root cause?
  - What data do you have to support this?
    - For example, metrics and graphs
  - What were the critical pillar implications, especially security?
    - When architecting workloads, you make trade-offs between pillars based upon your business context. These business decisions can drive your engineering priorities. You might optimize to reduce cost at the expense of reliability in development environments, or, for mission-critical solutions, you might optimize reliability with increased costs. Security is always job zero, as you have to protect your customers.
  - What lessons did you learn?
  - What corrective actions are you taking?

    - Action items
    - Related items
- Create well-defined standard operating procedures for conducting post-incident analyses.
- Set up a standardized incident reporting process. Document all incidents comprehensively, including the initial incident report, logs, communications, and actions taken during the incident.
- Remember that an incident does not require an outage. It could be a near-miss, or a system performing in an unexpected way while still fulfilling its business function.
- Continually improve your post-incident analysis process based on feedback and lessons learned.
- Capture key findings in a knowledge management system, and consider any patterns that should be added to developer guides or pre-deployment checklists.

Resources

Related documents:

- Why you should develop a correction of error (COE)

Related videos:

- Amazon's approach to failing successfully
- AWS re:Invent 2021 - Amazon Builders' Library: Operational Excellence at Amazon

REL12-BP03 Test scalability and performance requirements

Use techniques such as load testing to validate that the workload meets scaling and performance requirements.

In the cloud, you can create a production-scale test environment for your workload on demand. Instead of reliance on a scaled-down test environment, which could lead to inaccurate predictions of production behaviors, you can use the cloud to provision a test environment that closely mirrors your expected production environment. This environment helps you test in a more accurate simulation of the real-world conditions your application faces.

Alongside your performance testing efforts, it's essential to validate that your base resources, scaling settings, service quotas, and resiliency design operate as expected under load. This holistic

approach verifies that your application can reliably scale and perform as required, even under the most demanding conditions.

Desired outcome: Your workload maintains its expected behavior even while subject to peak load. You proactively address any performance-related issues that may arise as the application grows and evolves.

Common anti-patterns:

- You use test environments that do not closely match the production environment.
- You treat load testing as a separate, one-time activity rather than an integrated part of the deployment continuous integration (CI) pipeline.
- You don't define clear and measurable performance requirements, such as response time, throughput, and scalability targets.
- You perform tests with unrealistic or insufficient load scenarios, and you fail to test for peak loads, sudden spikes, and sustained high load.
- You don't stress test the workload by exceeding expected load limits.
- You use inadequate or inappropriate load testing and performance profiling tools.
- You lack comprehensive monitoring and alerting systems to track performance metrics and detect anomalies.

Benefits of establishing this best practice:

- Load testing helps you identify potential performance bottlenecks in your system before it goes into production. When you simulate production-level traffic and workloads, you can identify areas where your system may struggle to handle the load, such as slow response times, resource constraints, or system failures.
- As you test your system under various load conditions, you can better understand the resource requirements needed to support your workload. This information can help you make informed decisions about resource allocation and prevent over-provisioning or under-provisioning of resources.
- To identify potential failure points, you can observe how your workload performs under high load conditions. This information helps you improve your workload's reliability and resiliency by implementing fault-tolerance mechanisms, failover strategies, and redundancy measures, as appropriate.

- You identify and address performance issues early, which helps you avoid the costly consequences of system outages, slow response times, and dissatisfied users.
- Detailed performance data and profiling information collected during testing can help you troubleshoot performance-related issues that may arise in production. This can lead to faster incident response and resolution, which reduces the impact on users and your organization's operations.
- In certain industries, proactive performance testing can help your workload meet compliance standards, which reduces the risk of penalties or legal issues.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The first step is to define a comprehensive testing strategy that covers all aspects of scaling and performance requirements. To start, clearly define your workload's service-level objectives (SLOs) based on your business needs, such as throughput, latency histogram, and error rate. Next, design a suite of tests that can simulate various load scenarios that range from average usage to sudden spikes and sustained peak loads, and verify that the workload's behavior meets your SLOs. These tests should be automated and integrated into your continuous integration and deployment pipeline to catch performance regressions early in the development process.

To effectively test scaling and performance, invest in the right tools and infrastructure. This includes load testing tools that can generate realistic user traffic, performance profiling tools to identify bottlenecks, and monitoring solutions to track key metrics. Importantly, you should verify that your test environments closely match the production environment in terms of infrastructure and environment conditions to make your test results as accurate as possible. To make it easier to reliably replicate and scale production-like setups, use infrastructure as code and container-based applications.

Scaling and performance tests are an ongoing process, not a one-time activity. Implement comprehensive monitoring and alerting to track the application's performance in production, and use this data to continually refine your test strategies and optimization efforts. Regularly analyze performance data to identify emerging issues, test new scaling strategies, and implement optimizations to improve the application's efficiency and reliability. When you adopt an iterative approach and constantly learn from production data, you can verify that your application can adapt to variable user demands and maintain resiliency and optimal performance over time.

Implementation steps

1. Establish clear and measurable performance requirements, such as response time, throughput, and scalability targets. These requirements should be based on your workload's usage patterns, user expectations, and business needs.
2. Select and configure a load testing tool that can accurately mimic the load patterns and user behavior in your production environment.
3. Set up a test environment that closely matches the production environment, including infrastructure and environment conditions, to improve the accuracy of your test results.
4. Create a test suite that covers a wide range of scenarios, from average usage patterns to peak loads, rapid spikes, and sustained high loads. Integrate the tests into your continuous integration and deployment pipelines to catch performance regressions early in the development process.
5. Conduct load testing to simulate real-world user traffic and understand how your application behaves under different load conditions. To stress test your application, exceed the expected load and observe its behavior, such as response time degradation, resource exhaustion, or system failures, which helps identify the breaking point of your application and inform scaling strategies. Evaluate the scalability of your workload by incrementally increasing the load, and measure the performance impact to identify scaling limits and plan for future capacity needs.
6. Implement comprehensive monitoring and alerting to track performance metrics, detect anomalies, and initiate scaling actions or notifications when thresholds are exceeded.
7. Continually monitor and analyze performance data to identify areas for improvement. Iterate on your testing strategies and optimization efforts.

Resources

Related best practices:

- REL01-BP04 Monitor and manage quotas
- REL06-BP01 Monitor all components for the workload (Generation)
- REL06-BP03 Send notifications (Real-time processing and alarming))

Related documents:

- Load testing applications
- Distributed Load Testing on AWS

- Application Performance Monitoring
- Amazon EC2 Testing Policy

Related examples:

- Distributed Load Testing on AWS (GitHub)

Related tools:

- Amazon CodeGuru Profiler
- Amazon CloudWatch RUM
- Apache JMeter
- K6
- Vegeta
- Hey
- ab
- wrk
- Distributed Load Testing on AWS

REL12-BP04 Test resiliency using chaos engineering

Run chaos experiments regularly in environments that are in or as close to production as possible to understand how your system responds to adverse conditions.

Desired outcome:

The resilience of the workload is regularly verified by applying chaos engineering in the form of fault injection experiments or injection of unexpected load, in addition to resilience testing that validates known expected behavior of your workload during an event. Combine both chaos engineering and resilience testing to gain confidence that your workload can survive component failure and can recover from unexpected disruptions with minimal to no impact.

Common anti-patterns:

- Designing for resiliency, but not verifying how the workload functions as a whole when faults occur.

- Never experimenting under real-world conditions and expected load.
- Not treating your experiments as code or maintaining them through the development cycle.
- Not running chaos experiments both as part of your CI/CD pipeline, as well as outside of deployments.
- Neglecting to use past post-incident analyses when determining which faults to experiment with.

Benefits of establishing this best practice: Injecting faults to verify the resilience of your workload allows you to gain confidence that the recovery procedures of your resilient design will work in the case of a real fault.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Chaos engineering provides your teams with capabilities to continually inject real world disruptions (simulations) in a controlled way at the service provider, infrastructure, workload, and component level, with minimal to no impact to your customers. It allows your teams to learn from faults and observe, measure, and improve the resilience of your workloads, as well as validate that alerts fire and teams get notified in the case of an event.

When performed continually, chaos engineering can highlight deficiencies in your workloads that, if left unaddressed, could negatively affect availability and operation.

> **Note**
>
> Chaos engineering is the discipline of experimenting on a system in order to build confidence in the system's capability to withstand turbulent conditions in production. – Principles of Chaos Engineering

If a system is able to withstand these disruptions, the chaos experiment should be maintained as an automated regression test. In this way, chaos experiments should be performed as part of your systems development lifecycle (SDLC) and as part of your CI/CD pipeline.

To ensure that your workload can survive component failure, inject real world events as part of your experiments. For example, experiment with the loss of Amazon EC2 instances or failover of the primary Amazon RDS database instance, and verify that your workload is not impacted (or

only minimally impacted). Use a combination of component faults to simulate events that may be caused by a disruption in an Availability Zone.

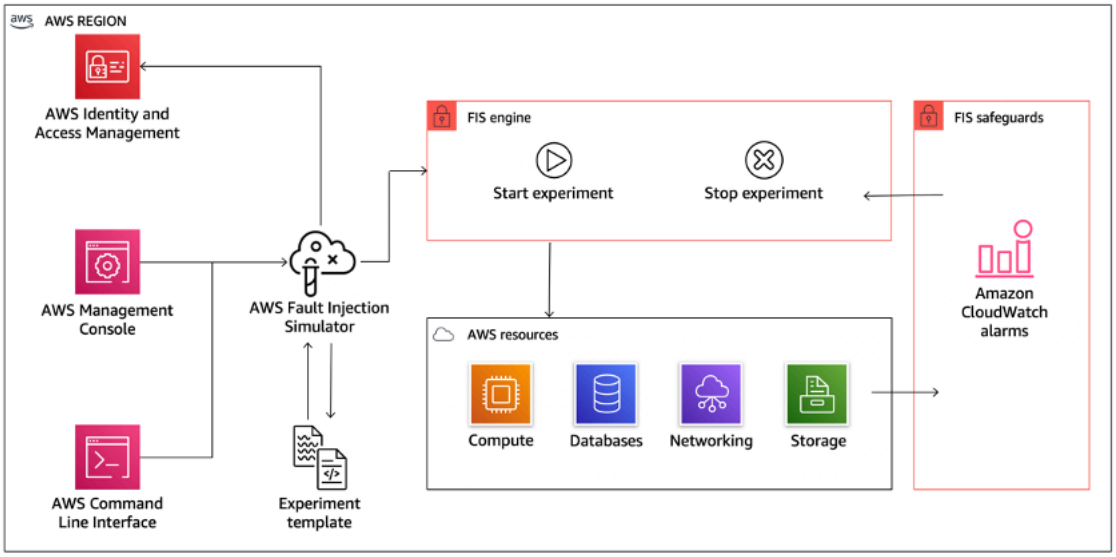
For application-level faults (such as crashes), you can start with stressors such as memory and CPU exhaustion.

To validate fallback or failover mechanisms for external dependencies due to intermittent network disruptions, your components should simulate such an event by blocking access to the third-party providers for a specified duration that can last from seconds to hours.

Other modes of degradation might cause reduced functionality and slow responses, often resulting in a disruption of your services. Common sources of this degradation are increased latency on critical services and unreliable network communication (dropped packets). Experiments with these faults, including networking effects such as latency, dropped messages, and DNS failures, could include the inability to resolve a name, reach the DNS service, or establish connections to dependent services.

Chaos engineering tools:

AWS Fault Injection Service (AWS FIS) is a fully managed service for running fault injection experiments that can be used as part of your CD pipeline, or outside of the pipeline. AWS FIS is a good choice to use during chaos engineering game days. It supports simultaneously introducing faults across different types of resources including Amazon EC2, Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and Amazon RDS. These faults include termination of resources, forcing failovers, stressing CPU or memory, throttling, latency, and packet loss. Since it is integrated with Amazon CloudWatch Alarms, you can set up stop conditions as guardrails to rollback an experiment if it causes unexpected impact.



AWS Fault Injection Service integrates with AWS resources to allow you to run fault injection experiments for your workloads.

There are also several third-party options for fault injection experiments. These include open-source tools such as Chaos Toolkit, Chaos Mesh, and Litmus Chaos, as well as commercial options like Gremlin. To expand the scope of faults that can be injected on AWS, AWS FIS integrates with Chaos Mesh and Litmus Chaos, allowing you to coordinate fault injection workflows among multiple tools. For example, you can run a stress test on a pod's CPU using Chaos Mesh or Litmus faults while terminating a randomly selected percentage of cluster nodes using AWS FIS fault actions.

Implementation steps

1. Determine which faults to use for experiments.

   Assess the design of your workload for resiliency. Such designs (created using the best practices of the Well-Architected Framework) account for risks based on critical dependencies, past events, known issues, and compliance requirements. List each element of the design intended to maintain resilience and the faults it is designed to mitigate. For more information about creating such lists, see the Operational Readiness Review whitepaper which guides you on how to create a process to prevent reoccurrence of previous incidents. The Failure Modes and Effects Analysis (FMEA) process provides you with a framework for performing a component-level analysis of failures and how they impact your workload. FMEA is outlined in more detail by Adrian Cockcroft in Failure Modes and Continuous Resilience.
2. Assign a priority to each fault.

Start with a coarse categorization such as high, medium, or low. To assess priority, consider frequency of the fault and impact of failure to the overall workload.

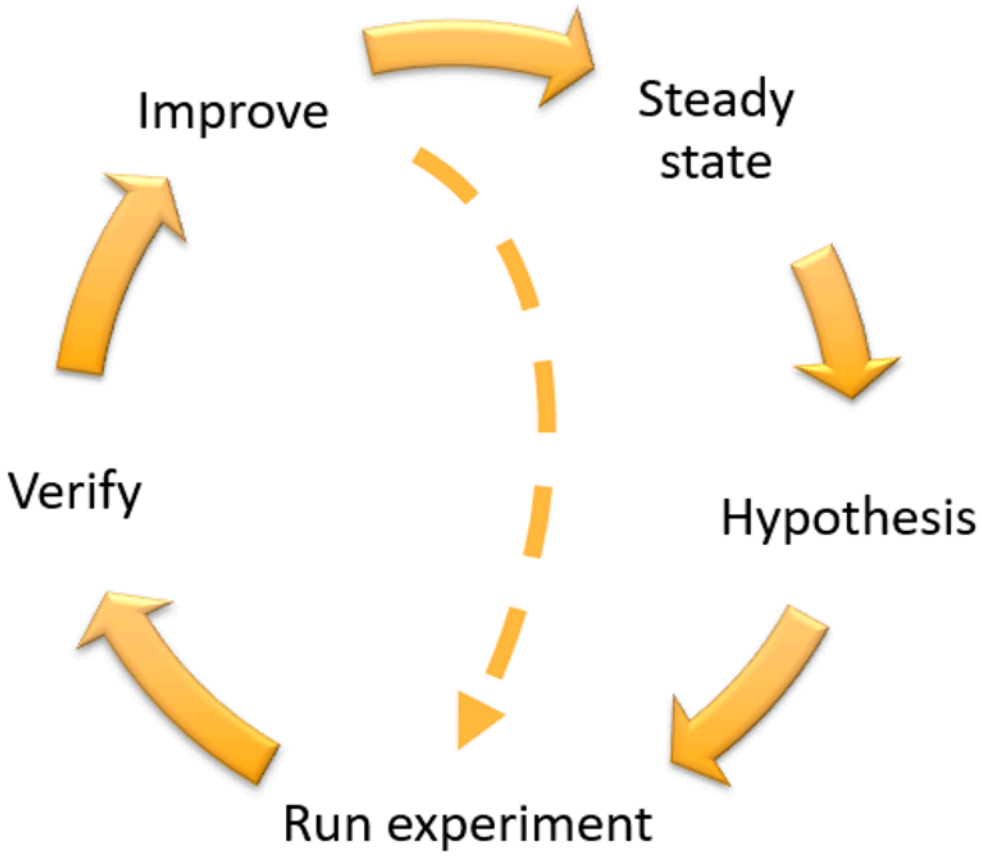
When considering frequency of a given fault, analyze past data for this workload when available. If not available, use data from other workloads running in a similar environment.

When considering impact of a given fault, the larger the scope of the fault, generally the larger the impact. Also consider the workload design and purpose. For example, the ability to access the source data stores is critical for a workload doing data transformation and analysis. In this case, you would prioritize experiments for access faults, as well as throttled access and latency insertion.

Post-incident analyses are a good source of data to understand both frequency and impact of failure modes.

Use the assigned priority to determine which faults to experiment with first and the order with which to develop new fault injection experiments.

3. For each experiment that you perform, follow the chaos engineering and continuous resilience flywheel in the following figure.



Chaos engineering and continuous resilience flywheel, using the scientific method by Adrian Hornsby.

a. Define steady state as some measurable output of a workload that indicates normal behavior.

   Your workload exhibits steady state if it is operating reliably and as expected. Therefore, validate that your workload is healthy before defining steady state. Steady state does not necessarily mean no impact to the workload when a fault occurs, as a certain percentage in faults could be within acceptable limits. The steady state is your baseline that you will observe during the experiment, which will highlight anomalies if your hypothesis defined in the next step does not turn out as expected.

   For example, a steady state of a payments system can be defined as the processing of 300 TPS with a success rate of 99% and round-trip time of 500 ms.

b. Form a hypothesis about how the workload will react to the fault.

A good hypothesis is based on how the workload is expected to mitigate the fault to maintain the steady state. The hypothesis states that given the fault of a specific type, the system or workload will continue steady state, because the workload was designed with specific mitigations. The specific type of fault and mitigations should be specified in the hypothesis.

The following template can be used for the hypothesis (but other wording is also acceptable):

> **Note**
>
> If *`specific fault`* occurs, the *`workload name`* workload will *`describe mitigating controls`* to maintain *`business or technical metric impact`*.

For example:

- If 20% of the nodes in the Amazon EKS node-group are taken down, the Transaction Create API continues to serve the 99th percentile of requests in under 100 ms (steady state). The Amazon EKS nodes will recover within five minutes, and pods will get scheduled and process traffic within eight minutes after the initiation of the experiment. Alerts will fire within three minutes.
- If a single Amazon EC2 instance failure occurs, the order system's Elastic Load Balancing health check will cause the Elastic Load Balancing to only send requests to the remaining healthy instances while the Amazon EC2 Auto Scaling replaces the failed instance, maintaining a less than 0.01% increase in server-side (5xx) errors (steady state).
- If the primary Amazon RDS database instance fails, the Supply Chain data collection workload will failover and connect to the standby Amazon RDS database instance to maintain less than 1 minute of database read or write errors (steady state).

c. Run the experiment by injecting the fault.

An experiment should by default be fail-safe and tolerated by the workload. If you know that the workload will fail, do not run the experiment. Chaos engineering should be used to find known-unknowns or unknown-unknowns. *Known-unknowns* are things you are aware of but don't fully understand, and *unknown-unknowns* are things you are neither aware of nor fully understand. Experimenting against a workload that you know is broken won't provide you with new insights. Your experiment should be carefully planned, have a clear scope of impact, and provide a rollback mechanism that can be applied in case of unexpected turbulence. If your due-diligence shows that your workload should survive the experiment, move forward

with the experiment. There are several options for injecting the faults. For workloads on AWS, AWS FIS provides many predefined fault simulations called actions. You can also define custom actions that run in AWS FIS using AWS Systems Manager documents.

We discourage the use of custom scripts for chaos experiments, unless the scripts have the capabilities to understand the current state of the workload, are able to emit logs, and provide mechanisms for rollbacks and stop conditions where possible.

An effective framework or toolset which supports chaos engineering should track the current state of an experiment, emit logs, and provide rollback mechanisms to support the controlled running of an experiment. Start with an established service like AWS FIS that allows you to perform experiments with a clearly defined scope and safety mechanisms that rollback the experiment if the experiment introduces unexpected turbulence. To learn about a wider variety of experiments using AWS FIS, also see the Resilient and Well-Architected Apps with Chaos Engineering lab. Also, AWS Resilience Hub will analyze your workload and create experiments that you can choose to implement and run in AWS FIS.

> **Note**
>
> For every experiment, clearly understand the scope and its impact. We recommend that faults should be simulated first on a non-production environment before being run in production.

Experiments should run in production under real-world load using canary deployments that spin up both a control and experimental system deployment, where feasible. Running experiments during off-peak times is a good practice to mitigate potential impact when first experimenting in production. Also, if using actual customer traffic poses too much risk, you can run experiments using synthetic traffic on production infrastructure against the control and experimental deployments. When using production is not possible, run experiments in pre-production environments that are as close to production as possible.

You must establish and monitor guardrails to ensure the experiment does not impact production traffic or other systems beyond acceptable limits. Establish stop conditions to stop an experiment if it reaches a threshold on a guardrail metric that you define. This should include the metrics for steady state for the workload, as well as the metric against the components into which you're injecting the fault. A synthetic monitor (also known as a user canary) is one metric you should usually include as a user proxy. Stop conditions for AWS FIS

are supported as part of the experiment template, allowing up to five stop-conditions per template.

One of the principles of chaos is minimize the scope of the experiment and its impact:

While there must be an allowance for some short-term negative impact, it is the responsibility and obligation of the Chaos Engineer to ensure the fallout from experiments are minimized and contained.

A method to verify the scope and potential impact is to perform the experiment in a non-production environment first, verifying that thresholds for stop conditions activate as expected during an experiment and observability is in place to catch an exception, instead of directly experimenting in production.

When running fault injection experiments, verify that all responsible parties are well-informed. Communicate with appropriate teams such as the operations teams, service reliability teams, and customer support to let them know when experiments will be run and what to expect. Give these teams communication tools to inform those running the experiment if they see any adverse effects.

You must restore the workload and its underlying systems back to the original known-good state. Often, the resilient design of the workload will self-heal. But some fault designs or failed experiments can leave your workload in an unexpected failed state. By the end of the experiment, you must be aware of this and restore the workload and systems. With AWS FIS you can set a rollback configuration (also called a post action) within the action parameters. A post action returns the target to the state that it was in before the action was run. Whether automated (such as using AWS FIS) or manual, these post actions should be part of a playbook that describes how to detect and handle failures.

d. Verify the hypothesis.

Principles of Chaos Engineering gives this guidance on how to verify steady state of your workload:

Focus on the measurable output of a system, rather than internal attributes of the system. Measurements of that output over a short period of time constitute a proxy for the system's steady state. The overall system's throughput, error rates, and latency percentiles could all be metrics of interest representing steady state behavior. By focusing on systemic behavior patterns during experiments, chaos engineering verifies that the system does work, rather than trying to validate how it works.

In our two previous examples, we include the steady state metrics of less than 0.01% increase in server-side (5xx) errors and less than one minute of database read and write errors.

The 5xx errors are a good metric because they are a consequence of the failure mode that a client of the workload will experience directly. The database errors measurement is good as a direct consequence of the fault, but should also be supplemented with a client impact measurement such as failed customer requests or errors surfaced to the client. Additionally, include a synthetic monitor (also known as a user canary) on any APIs or URIs directly accessed by the client of your workload.

e. Improve the workload design for resilience.

If steady state was not maintained, then investigate how the workload design can be improved to mitigate the fault, applying the best practices of the AWS Well-Architected Reliability pillar. Additional guidance and resources can be found in the AWS Builder's Library, which hosts articles about how to improve your health checks or employ retries with backoff in your application code, among others.

After these changes have been implemented, run the experiment again (shown by the dotted line in the chaos engineering flywheel) to determine their effectiveness. If the verify step indicates the hypothesis holds true, then the workload will be in steady state, and the cycle continues.

4. Run experiments regularly.

A chaos experiment is a cycle, and experiments should be run regularly as part of chaos engineering. After a workload meets the experiment's hypothesis, the experiment should be automated to run continually as a regression part of your CI/CD pipeline. To learn how to do this, see this blog on how to run AWS FIS experiments using AWS CodePipeline. This lab on recurrent AWS FIS experiments in a CI/CD pipeline allows you to work hands-on.

Fault injection experiments are also a part of game days (see REL12-BP05 Conduct game days regularly). Game days simulate a failure or event to verify systems, processes, and team responses. The purpose is to actually perform the actions the team would perform as if an exceptional event happened.

5. Capture and store experiment results.

Results for fault injection experiments must be captured and persisted. Include all necessary data (such as time, workload, and conditions) to be able to later analyze experiment results and

trends. Examples of results might include screenshots of dashboards, CSV dumps from your metric's database, or a hand-typed record of events and observations from the experiment. Experiment logging with AWS FIS can be part of this data capture.

Resources

Related best practices:

- REL08-BP03 Integrate resiliency testing as part of your deployment
- REL13-BP03 Test disaster recovery implementation to validate the implementation

Related documents:

- What is AWS Fault Injection Service?
- What is AWS Resilience Hub?
- Principles of Chaos Engineering
- Chaos Engineering: Planning your first experiment
- Resilience Engineering: Learning to Embrace Failure
- Chaos Engineering stories
- Avoiding fallback in distributed systems
- Canary Deployment for Chaos Experiments

Related videos:

- AWS re:Invent 2020: Testing resiliency using chaos engineering (ARC316)
- AWS re:Invent 2019: Improving resiliency with chaos engineering (DOP309-R1)
- AWS re:Invent 2019: Performing chaos engineering in a serverless world (CMY301)

Related tools:

- AWS Fault Injection Service
- AWS Marketplace: Gremlin Chaos Engineering Platform
- Chaos Toolkit
- Chaos Mesh

- Litmus

REL12-BP05 Conduct game days regularly

Conduct game days to regularly exercise your procedures for responding to workload-impacting events and impairments. Involve the same teams who would be responsible for handling production scenarios. These exercises help enforce measures to prevent user impact caused by production events. When you practice your response procedures in realistic conditions, you can identify and address any gaps or weaknesses before a real event occurs.

Game days simulate events in production-like environments to test systems, processes, and team responses. The purpose is to perform the same actions the team would perform as if the event actually occurred. These exercises help you understand where improvements can be made and can help develop organizational experience in dealing with events and impairments. These should be conducted regularly so that your team knows builds ingrained habits for how to respond.

Game days prepare teams to handle production events with greater confidence. Teams that are well-practiced are more able to quickly detect and respond to various scenarios. This results in a significantly improved readiness and resilience posture.

Desired outcome: You run resilience game days on a consistent, scheduled basis. These game days are seen as a normal and expected part of doing business. Your organization has built a culture of preparedness, and when production issues occur, your teams are well-prepared to respond effectively, resolve the issues efficiently, and mitigate the impact on customers.

Common anti-patterns:

- You document your procedures, but your never exercise them.
- You exclude business decision makers in the test exercises.
- You run a game day, but you don't inform all relevant stakeholders.
- You focus solely on technical failures, but you don't involve business stakeholders.
- You don't incorporate lessons learned from game days into your recovery processes.
- You blame teams for failures or bugs.

Benefits of establishing this best practice:

- Enhance response skills: On game days, teams practice their duties and test their communication mechanisms during simulated events, which creates a more coordinated and efficient response in production situations.
- Identify and address dependencies: Complex environments often involve intricate dependencies between various systems, services, and components. Game days can help you identify and address these dependencies, and verify that your critical systems and services are properly covered by your runbook procedures and can be scaled up or recovered in a timely manner.
- Foster a culture of resilience: Game days can help cultivate a mindset of resilience within an organization. When you involve cross-functional teams and stakeholders, these exercises promote awareness, collaboration, and a shared understanding of the importance of resilience across the entire organization.
- Continuous improvement and adaptation: Regular game days help you to continually assess and adapt your resilience strategies, which keeps them relevant and effective in the face of changing circumstances.
- Increase confidence in the system: Successful game days can help you build confidence in the system's ability to withstand and recover from disruptions.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Once you have designed and implemented the necessary resilience measures, conduct a game day to validate that everything works as planned in production. A game day, especially the first one, should involve all team members, and all stakeholders and participants should be informed in advance about the date, time, and simulated scenarios.

During the game day, the involved teams simulate various events and potential scenarios according to the prescribed procedures. The participants closely monitor and assess the impact of these simulated events. If the system operates as designed, the automated detection, scaling, and self-healing mechanisms should activate and result in little to no impact on users. If the team observes any negative impact, they roll back the test and remedy the identified issues, either through automated means or manual intervention documented in the applicable runbooks.

To continuously improve resilience, it's critical to document and incorporate lessons learned. This process is a *feedback loop* that systematically captures insights from game days and uses them to enhance systems, processes, and team capabilities.

To help you reproduce real-world scenarios where system components or services may fail unexpectedly, inject simulated faults as a game day exercise. Teams can test the resilience and fault tolerance of their systems and simulate their incident response and recovery processes in a controlled environment.

In AWS, your game days can be carried out with replicas of your production environment using infrastructure as code. Through this process, you can test in a safe environment that closely resembles your production environment. Consider AWS Fault Injection Service to create different failure scenarios. Use services like Amazon CloudWatch and AWS X-Ray to monitor system behavior during game days. Use AWS Systems Manager to manage and run playbooks, and use AWS Step Functions to orchestrate recurring game day workflows.

Implementation steps

- **Establish a game day program:** Develop a structured program that defines the frequency, scope and objectives of game days. Involve key stakeholders and subject matter experts in planning and running these exercises.
- **Prepare the game day:**
  1. Identify the key business-critical services that are the focus of the game day. Catalog and map the people, processes, and technologies that support those services.
  2. Set the agenda for the game day, and prepare the involved teams to participate in the event. Prepare your automation services to simulate the planned scenarios and run the appropriate recovery processes. AWS services such as AWS Fault Injection Service, AWS Step Functions, and AWS Systems Manager can help you automate various aspects of game days, such as injection of faults and initiation of recovery actions.
- **Run your simulation:** On the game day, run the planned scenario. Observe and document how the people, processes, and technologies react to the simulated event.
- **Conduct post-exercise reviews:** After the game day, conduct a retrospective session to review the lessons learned. Identify areas for improvement and any actions needed to improve operational resilience. Document your findings, and track any necessary changes to enhance your resilience strategies and preparedness to completion.

Resources

Related best practices:

- REL12-BP01 Use playbooks to investigate failures

- REL12-BP04 Test resiliency using chaos engineering
- OPS04-BP01 Identify key performance indicators
- OPS07-BP03 Use runbooks to perform procedures
- OPS10-BP01 Use a process for event, incident, and problem management

Related documents:

- What is AWS GameDay?

Related videos:

- AWS re:Invent 2023 - Practice like you play: How Amazon scales resilience to new heights

Related examples:

- AWS Workshop - Navigate the storm: Unleashing controlled chaos for resilient systems
- Build Your Own Game Day to Support Operational Resilience

REL 13. How do you plan for disaster recovery (DR)?

Having backups and redundant workload components in place is the start of your DR strategy. RTO and RPO are your objectives for restoration of your workload. Set these based on business needs. Implement a strategy to meet these objectives, considering locations and function of workload resources and data. The probability of disruption and cost of recovery are also key factors that help to inform the business value of providing disaster recovery for a workload.

Best practices

- REL13-BP01 Define recovery objectives for downtime and data loss
- REL13-BP02 Use defined recovery strategies to meet the recovery objectives
- REL13-BP03 Test disaster recovery implementation to validate the implementation
- REL13-BP04 Manage configuration drift at the DR site or Region
- REL13-BP05 Automate recovery

REL13-BP01 Define recovery objectives for downtime and data loss

Failures can impact your business in several ways. First, failures can cause service interruption (downtime). Second, failures can cause data to become lost, inconsistent, or stale. In order to guide how you respond and recover from failures, define a Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for each workload. *Recovery Time Objective (RTO)* is the maximum acceptable delay between the interruption of service and restoration of service. *Recovery Point Objective (RPO)* is the maximum acceptable time after the last data recovery point.

Desired outcome: Every workload has a designated RTO and RPO based on technical considerations and business impact.

Common anti-patterns:

- You haven't designated recovery objectives.
- You select arbitrary recovery objectives.
- You select recovery objectives that are too lenient and do not meet business objectives.
- You have not evaluated the impact of downtime and data loss.
- You select unrealistic recovery objectives, such as zero time to recover or zero data loss, which may not be achievable for your workload configuration.
- You select recovery objectives that are more stringent than actual business objectives. This forces recovery implementations that are costlier and more complicated than what the workload needs.
- You select recovery objectives that are incompatible with those of a dependent workload.
- You fail to consider regulatory and compliance requirements.

Benefits of establishing this best practice: When you set RTOs and RPOs for your workloads, you establish clear and measurable goals for recovery based on your business needs. Once you've set those goals, you can create disaster recovery (DR) plans that are tailored to meet them.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Construct a matrix or worksheet to help guide your disaster recovery planning. In your matrix, create different workload categories or tiers based on their business impact (such as critical, high, medium, and low) and the associated RTOs and RPOs to target for each one. The following matrix provides an example (note that your RTO and RPO values may differ) you can follow:

Disaster Recovery Matrix						
		Recovery Point Objective				
		< 1 Minute	< 1 Hour	< 6 Hours	< 1 Day	+ 1 Day
Recovery Time Objective	< 10 Minutes	Critical	Critical	High	Medium	Medium
	< 2 Hours	Critical	High	Medium	Medium	Low
	< 8 Hours	High	Medium	Medium	Low	Low
	< 24 Hours	Medium	Medium	Low	Low	Low
	24 + Hours	Medium	Low	Low	Low	Low

Example disaster recovery matrix

For each workload, investigate and understand the impact of downtime and lost data on your business. The impact typically grows with downtime and data loss, but the shape of the impact can differ based on the workload type. For example, downtime for up to an hour might have low impact, but after that, the impact could quickly intensify. Impact can take many forms, including financial impact (such as lost revenue), reputational impact (including loss of customer trust), operational impact (such as a missed payroll or decreased productivity), and regulatory risk. Once completed, assign the workload to the appropriate tier.

Consider the following questions when you analyze the impact of failure:

1. What is the maximum time the workload can be unavailable before unacceptable impact to the business is incurred?
2. How much impact, and what kind, will be incurred by the business by a workload disruption? Consider all kinds of impact, including financial, reputational, operational, and regulatory.
3. What is the maximum amount of data that can be lost or unrecoverable before unacceptable impact to the business is incurred?
4. Can lost data be recreated from other sources (also known as *derived* data)? If so, also consider the RPOs of all source data used to recreate the workload data.
5. What are the recovery objectives and availability expectations of workloads that this one depends on (downstream)? Your workload's objectives must be achievable given the recovery capabilities of its downstream dependencies. Consider possible downstream dependency workarounds or mitigations that can improve this workload's recovery capability.

6. What are the recovery objectives and availability expectations of workloads that depend on this one (upstream)? Upstream workload objectives may require this workload to have more stringent recovery capabilities than it first appears.
7. Are there different recovery objectives based on the type of incident? For example, you might have different RTOs and RPOs depending on whether the incident impacts an Availability Zone or an entire Region.
8. Do your recovery objectives change during certain events or times of the year? For example, you might have different RTOs and RPOs around holiday shopping seasons, sporting events, special sales, and new product launches.
9. How do the recovery objectives align with any line of business and organizational disaster recovery strategy you might have?
10. Are there legal or contractual ramifications to consider? For example, are you contractually obligated to provide a service with a given RTO or RPO? What penalties might you incur for not meeting them?
11. Are you required to maintain data integrity to meet regulatory or compliance requirements?

The following worksheet can aid your evaluation of each workload. You may modify this worksheet to suit your specific needs, such as adding additional questions.

Step 2: Primary questions	Applies to workload?	workload RTO	workload RPO	RTO adjust.	RPO adjust.	Instructions
[1] maximum time the workload can be down						measured in time from start of outage to recovery
[2] maximum amount of data that can be lost						measured in time since last known good restorable dataset
[3a] upstream dependencies						enter the most strict upstream recovery objectives
[3b] downstream dependencies						enter the least strict downstream recovery objectives
[3a] reconciled upstream dependencies						If upstream value is less then current values and downstream value greater,
[3b] reconciled downstream dependencies						then work with dependencies to reconcile and enter reconciled values here
[3] dependencies						lower values to meet upstream dependencies or raise them based on downstream dependency capabilities
Step 2: Additional questions						Indicate if question applies. If it does not apply then skip it
Base RTO/RPO						Carry RTO and RPO values from above down to here
[4] type of outage	[ ]Y / [ ]N					Enter recovery objectives for event type with strictest requirements
[5] specific time-based objectives	[ ]Y / [ ]N					Enter recovery objectives for times with the strictest requirements
[6] customers disrupted	[ ]Y / [ ]N					Graph customers impacted as a function of time down or data lost. Use that to enter the maximum RTO and RPO permissible based on customer impact
[7] reputation impact	[ ]Y / [ ]N					Work with the business to determine maximum RTO and RPO based on impact to reputation
[8] operational impact	[ ]Y / [ ]N					Enter maximum RTO and RPO based on operational impact
[9] organizational alignment	[ ]Y / [ ]N					Enter maximum RTO and RPO for workloads of this type as per LOB and organizational requirements
[10] contractual obligations	[ ]Y / [ ]N					Enter maximum RTO and RPO based on contractual obligations
[11] regulatory compliance	[ ]Y / [ ]N					Enter maximum RTO and RPO based on applicable regulatory compliance
target based on additional questions						Take the minimum value (stricter value) from Q's 4-11 and enter it here
adjusted target						If the objectives on the above line cannot be accommodated, work with stakeholders to loosen constraints, and enter new minimum here
Adjusted RTO/RPO						Enter base RPO/RTO values, or adjusted target, whichever is lower
Step 3						
Map to predefined category or tier						Adjust both values to downward (more strict) to align to nearest defined tier

Worksheet

Implementation steps

1. Identify the business stakeholders and technical teams responsible for each workload, and engage with them.
2. Create categories or tiers of criticality for workload impact in your organization. Example categories include critical, high, medium, and low. For each category, choose an RTO and RPO that reflects your business objectives and requirements.
3. Assign one of the impact categories you created in the previous step to each workload. To decide how a workload maps to a category, consider the workload's importance to the business and the impact of interruption or data loss, and use the questions above to guide you. This results in an RTO and RPO for each workload.
4. Consider the RTO and RPO for each workload determined in the previous step. Involve the workload's business and technical teams to determine whether the objectives should be adjusted. For example, business stakeholders could determine that more stringent targets are required. Alternatively, technical teams could determine that targets should be modified to make them achievable with available resources and technological constraints.

Resources

Related best practices:

- REL09-BP04 Perform periodic recovery of the data to verify backup integrity and processes
- REL12-BP01 Use playbooks to investigate failures
- REL13-BP02 Use defined recovery strategies to meet the recovery objectives
- REL13-BP03 Test disaster recovery implementation to validate the implementation

Related documents:

- AWS Architecture Blog: Disaster Recovery Series
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- Managing resiliency policies with AWS Resilience Hub
- APN Partner: partners that can help with disaster recovery
- AWS Marketplace: products that can be used for disaster recovery

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- Disaster Recovery of Workloads on AWS

REL13-BP02 Use defined recovery strategies to meet the recovery objectives

Define a disaster recovery (DR) strategy that meets your workload's recovery objectives. Choose a strategy such as backup and restore, standby (active/passive), or active/active.

Desired outcome: For each workload, there is a defined and implemented DR strategy that allows the workload to achieve DR objectives. DR strategies between workloads make use of reusable patterns (such as the strategies previously described),

Common anti-patterns:

- Implementing inconsistent recovery procedures for workloads with similar DR objectives.
- Leaving the DR strategy to be implemented ad-hoc when a disaster occurs.
- Having no plan for disaster recovery.

- Dependency on control plane operations during recovery.

Benefits of establishing this best practice:

- Using defined recovery strategies allows you to use common tooling and test procedures.
- Using defined recovery strategies improves knowledge sharing between teams and implementation of DR on the workloads they own.

Level of risk exposed if this best practice is not established: High. Without a planned, implemented, and tested DR strategy, you are unlikely to achieve recovery objectives in the event of a disaster.

Implementation guidance

A DR strategy relies on the ability to stand up your workload in a recovery site if your primary location becomes unable to run the workload. The most common recovery objectives are RTO and RPO, as discussed in REL13-BP01 Define recovery objectives for downtime and data loss.

A DR strategy across multiple Availability Zones (AZs) within a single AWS Region, can provide mitigation against disaster events like fires, floods, and major power outages. If it is a requirement to implement protection against an unlikely event that prevents your workload from being able to run in a given AWS Region, you can use a DR strategy that uses multiple Regions.

When architecting a DR strategy across multiple Regions, you should choose one of the following strategies. They are listed in increasing order of cost and complexity, and decreasing order of RTO and RPO. *Recovery Region* refers to an AWS Region other than the primary one used for your workload.

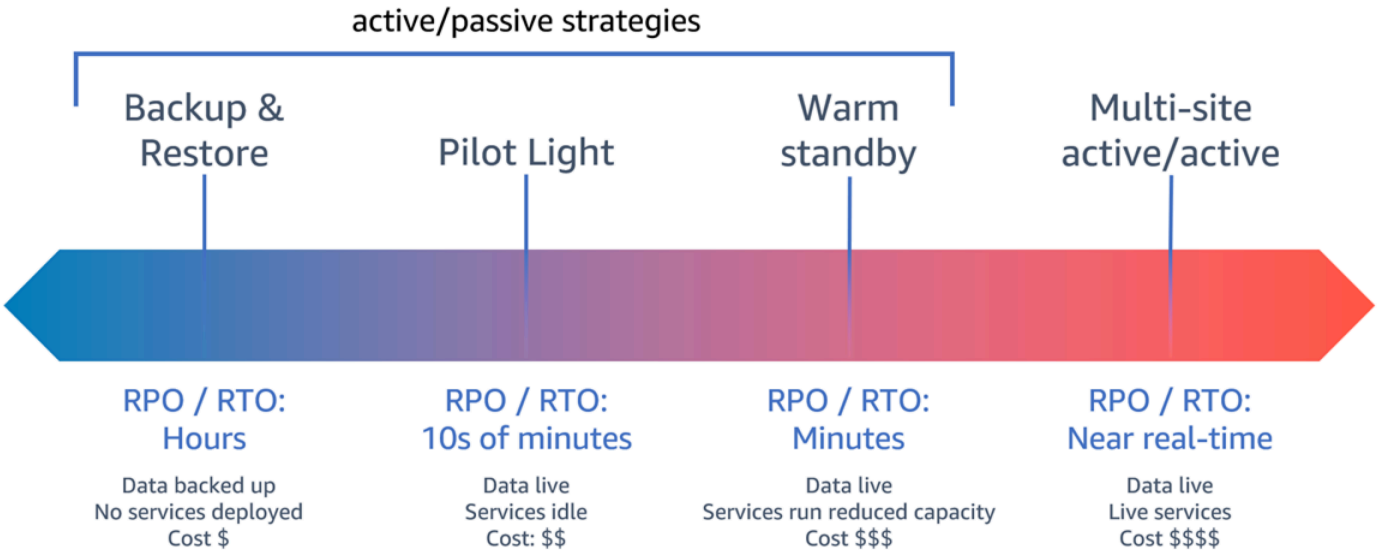


Figure 17: Disaster recovery (DR) strategies

- **Backup and restore** (RPO in hours, RTO in 24 hours or less): Back up your data and applications into the recovery Region. Using automated or continuous backups will permit point in time recovery (PITR), which can lower RPO to as low as 5 minutes in some cases. In the event of a disaster, you will deploy your infrastructure (using infrastructure as code to reduce RTO), deploy your code, and restore the backed-up data to recover from a disaster in the recovery Region.
- **Pilot light** (RPO in minutes, RTO in tens of minutes): Provision a copy of your core workload infrastructure in the recovery Region. Replicate your data into the recovery Region and create backups of it there. Resources required to support data replication and backup, such as databases and object storage, are always on. Other elements such as application servers or serverless compute are not deployed, but can be created when needed with the necessary configuration and application code.
- **Warm standby** (RPO in seconds, RTO in minutes): Maintain a scaled-down but fully functional version of your workload always running in the recovery Region. Business-critical systems are fully duplicated and are always on, but with a scaled down fleet. Data is replicated and live in the recovery Region. When the time comes for recovery, the system is scaled up quickly to handle the production load. The more scaled-up the warm standby is, the lower RTO and control plane reliance will be. When fully scales this is known as *hot standby*.
- **Multi-Region (multi-site) active-active** (RPO near zero, RTO potentially zero): Your workload is deployed to, and actively serving traffic from, multiple AWS Regions. This strategy requires you

to synchronize data across Regions. Possible conflicts caused by writes to the same record in two different regional replicas must be avoided or handled, which can be complex. Data replication is useful for data synchronization and will protect you against some types of disaster, but it will not protect you against data corruption or destruction unless your solution also includes options for point-in-time recovery.

> **Note**
>
> The difference between pilot light and warm standby can sometimes be difficult to understand. Both include an environment in your recovery Region with copies of your primary region assets. The distinction is that pilot light cannot process requests without additional action taken first, while warm standby can handle traffic (at reduced capacity levels) immediately. Pilot light will require you to turn on servers, possibly deploy additional (non-core) infrastructure, and scale up, while warm standby only requires you to scale up (everything is already deployed and running). Choose between these based on your RTO and RPO needs.
> When cost is a concern, and you wish to achieve a similar RPO and RTO objectives as defined in the warm standby strategy, you could consider cloud native solutions, like AWS Elastic Disaster Recovery, that take the pilot light approach and offer improved RPO and RTO targets.

Implementation steps

1. **Determine a DR strategy that will satisfy recovery requirements for this workload.**

   Choosing a DR strategy is a trade-off between reducing downtime and data loss (RTO and RPO) and the cost and complexity of implementing the strategy. You should avoid implementing a strategy that is more stringent than it needs to be, as this incurs unnecessary costs.

   For example, in the following diagram, the business has determined their maximum permissible RTO as well as the limit of what they can spend on their service restoration strategy. Given the business' objectives, the DR strategies pilot light or warm standby will satisfy both the RTO and the cost criteria.

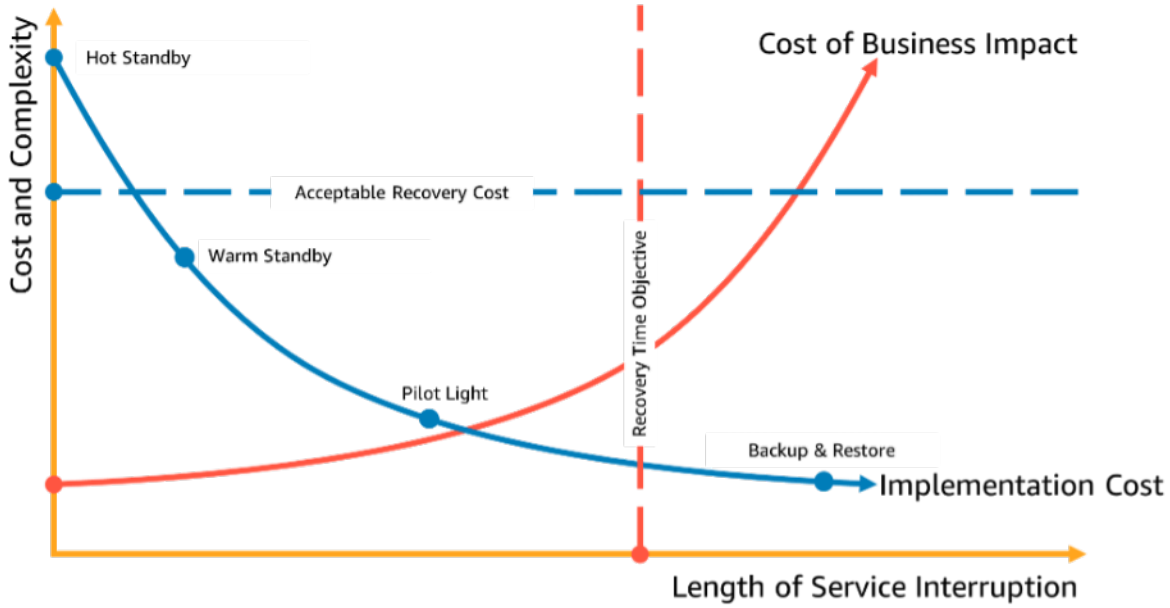


Figure 18: Choosing a DR strategy based on RTO and cost

To learn more, see Business Continuity Plan (BCP).

2. **Review the patterns for how the selected DR strategy can be implemented.**

   This step is to understand how you will implement the selected strategy. The strategies are explained using AWS Regions as the primary and recovery sites. However, you can also choose to use Availability Zones within a single Region as your DR strategy, which makes use of elements of multiple of these strategies.

   In the following steps, you can apply the strategy to your specific workload.

   Backup and restore

   Backup and restore is the least complex strategy to implement, but will require more time and effort to restore the workload, leading to higher RTO and RPO. It is a good practice to always make backups of your data, and copy these to another site (such as another AWS Region).

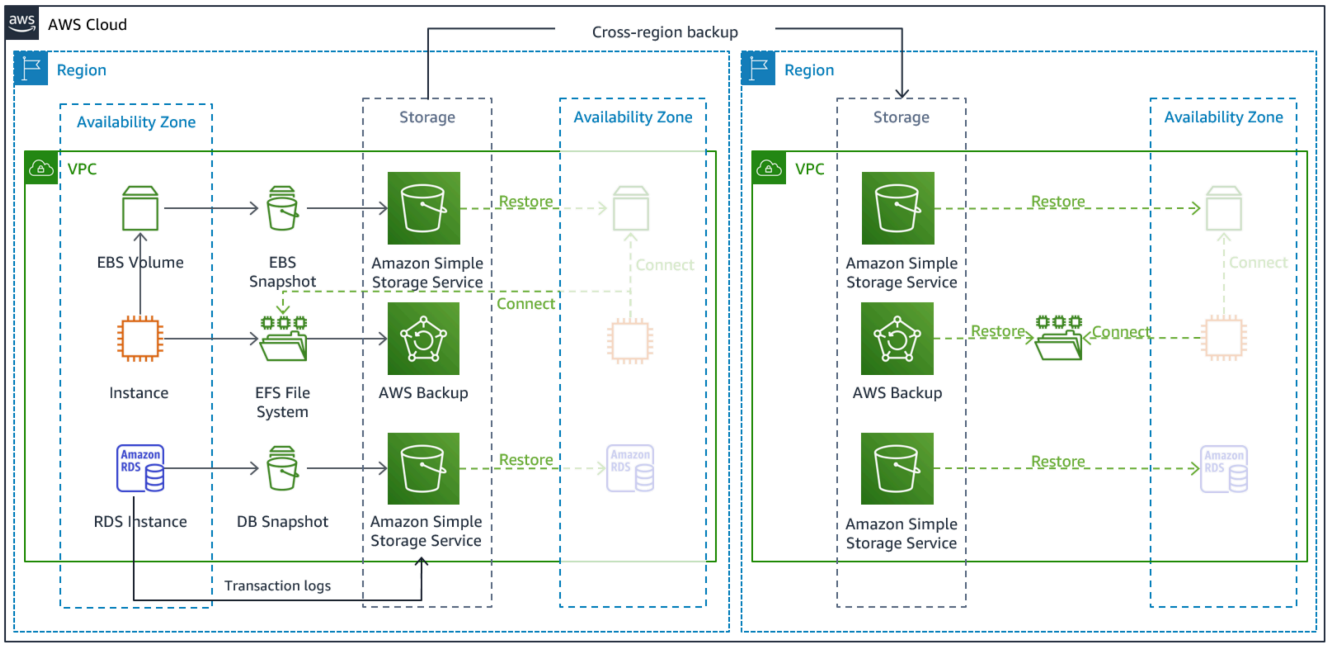


Figure 19: Backup and restore architecture

For more details on this strategy see [Disaster Recovery (DR) Architecture on AWS, Part II: Backup and Restore with Rapid Recovery].

Pilot light

With the *pilot light* approach, you replicate your data from your primary Region to your recovery Region. Core resources used for the workload infrastructure are deployed in the recovery Region, however additional resources and any dependencies are still needed to make this a functional stack. For example, in Figure 20, no compute instances are deployed.

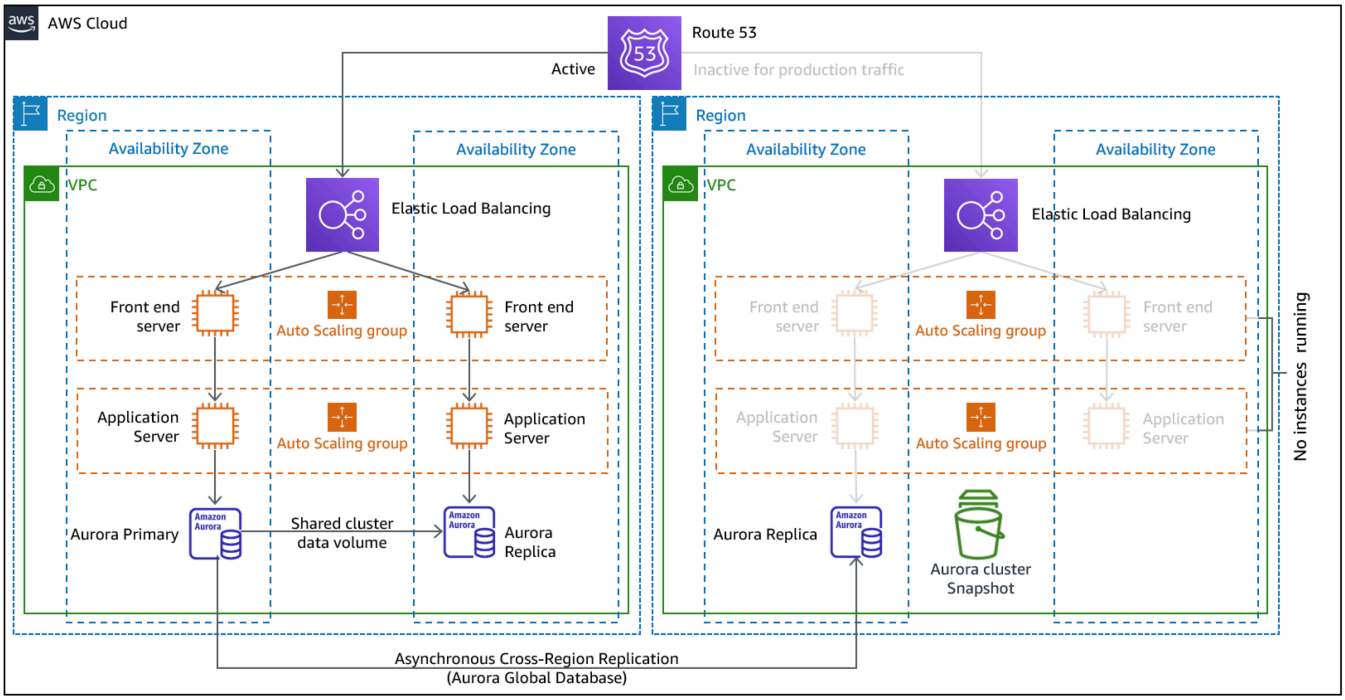


Figure 20: Pilot light architecture

For more details on this strategy, see Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby.

Warm standby

The *warm standby* approach involves ensuring that there is a scaled down, but fully functional, copy of your production environment in another Region. This approach extends the pilot light concept and decreases the time to recovery because your workload is always-on in another Region. If the recovery Region is deployed at full capacity, then this is known as *hot standby*.

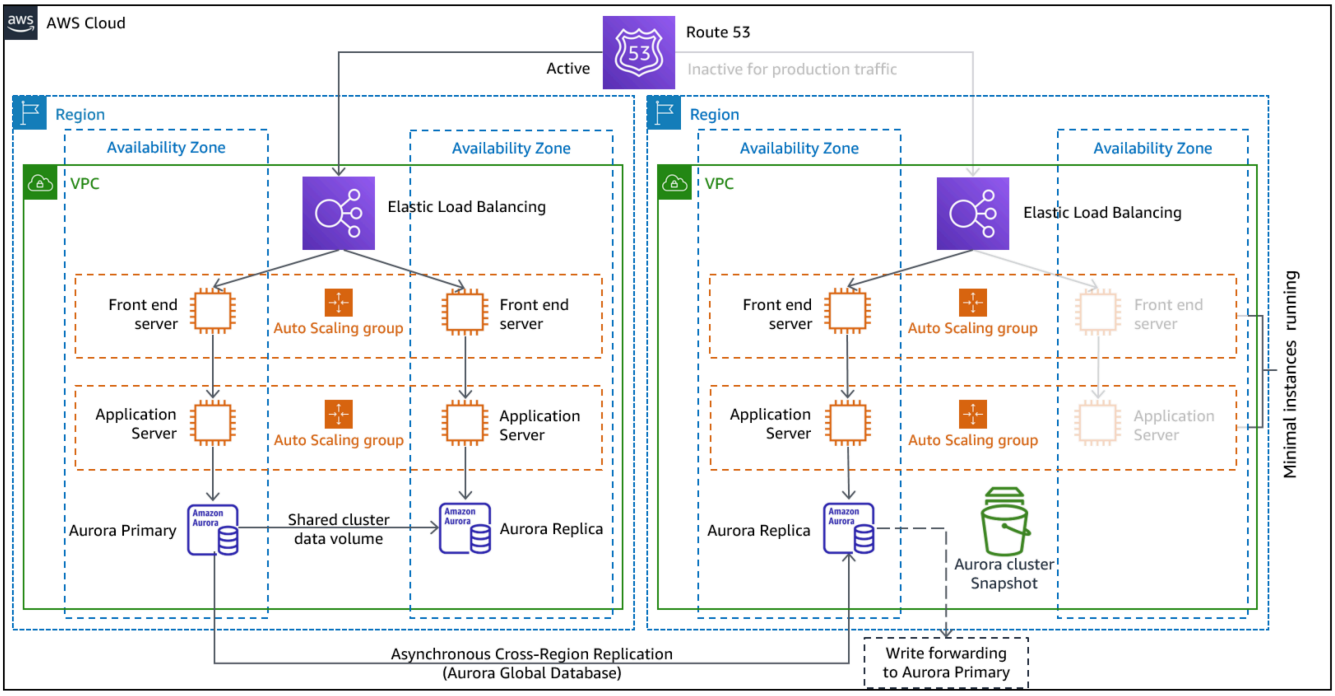


Figure 21: Warm standby architecture

Using warm standby or pilot light requires scaling up resources in the recovery Region. To verify capacity is available when needed, consider the use for capacity reservations for EC2 instances. If using AWS Lambda, then provisioned concurrency can provide runtime environments so that they are prepared to respond immediately to your function's invocations.

For more details on this strategy, see Disaster Recovery (DR) Architecture on AWS, Part III: Pilot Light and Warm Standby.

Multi-site active/active

You can run your workload simultaneously in multiple Regions as part of a *multi-site active/active* strategy. Multi-site active/active serves traffic from all regions to which it is deployed. Customers may select this strategy for reasons other than DR. It can be used to increase availability, or when deploying a workload to a global audience (to put the endpoint closer to users and/or to deploy stacks localized to the audience in that region). As a DR strategy, if the workload cannot be supported in one of the AWS Regions to which it is deployed, then that Region is evacuated, and the remaining Regions are used to maintain availability. Multi-site active/active is the most operationally complex of the DR strategies, and should only be selected when business requirements necessitate it.

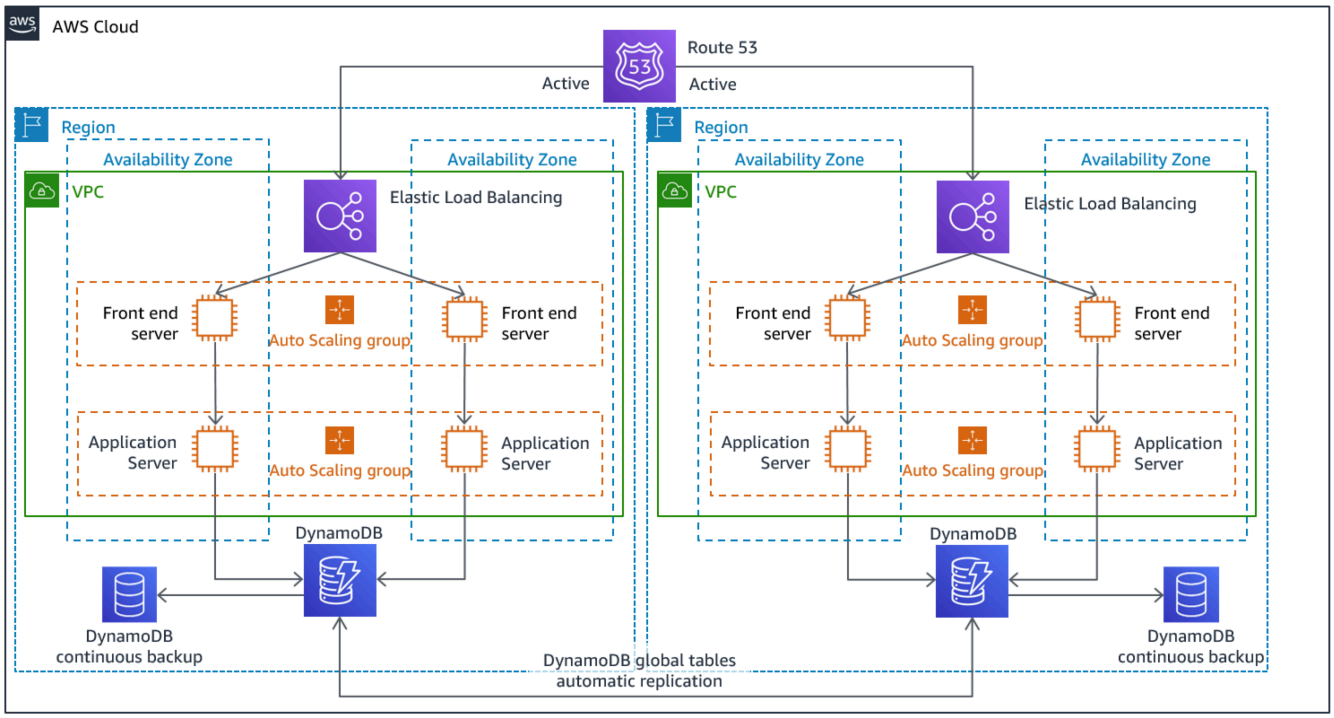


Figure 22: Multi-site active/active architecture

For more details on this strategy, see Disaster Recovery (DR) Architecture on AWS, Part IV: Multi-site Active/Active.

AWS Elastic Disaster Recovery

If you are considering the pilot light or warm standby strategy for disaster recovery, AWS Elastic Disaster Recovery could provide an alternative approach with improved benefits. Elastic Disaster Recovery can offer an RPO and RTO target similar to warm standby, but maintain the low-cost approach of pilot light. Elastic Disaster Recovery replicates your data from your primary region to your recovery Region, using continual data protection to achieve an RPO measured in seconds and an RTO that can be measured in minutes. Only the resources required to replicate the data are deployed in the recovery region, which keeps costs down, similar to the pilot light strategy. When using Elastic Disaster Recovery, the service coordinates and orchestrates the recovery of compute resources when initiated as part of failover or drill.

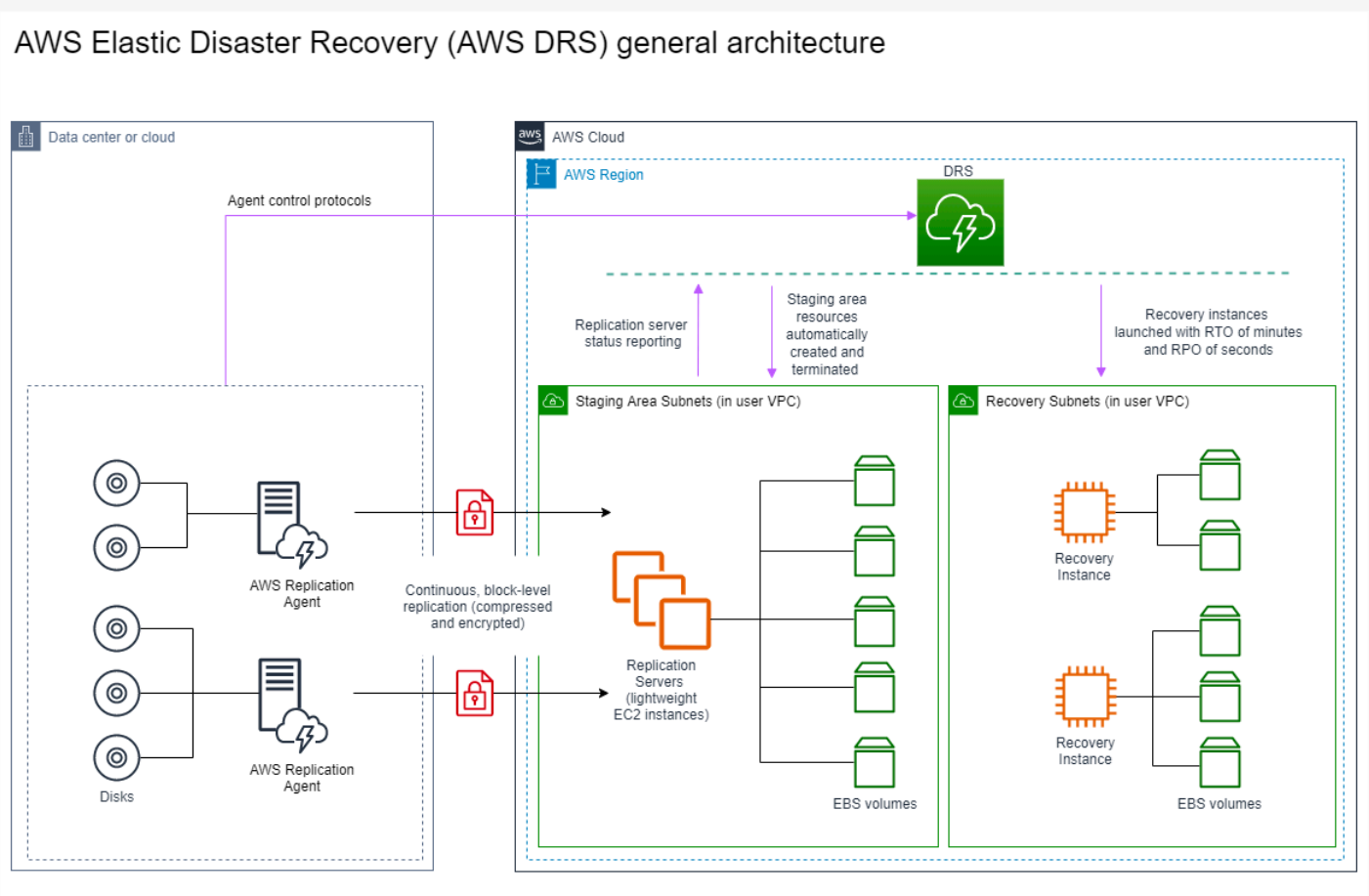


Figure 23: AWS Elastic Disaster Recovery architecture

Additional practices for protecting data

With all strategies, you must also mitigate against a data disaster. Continuous data replication protects you against some types of disaster, but it may not protect you against data corruption or destruction unless your strategy also includes versioning of stored data or options for point-in-time recovery. You must also back up the replicated data in the recovery site to create point-in-time backups in addition to the replicas.

Using multiple Availability Zones (AZs) within a single AWS Region

When using multiple AZs within a single Region, your DR implementation uses multiple elements of the above strategies. First you must create a high-availability (HA) architecture, using multiple AZs as shown in Figure 23. This architecture makes use of a multi-site active/active approach, as the Amazon EC2 instances and the Elastic Load Balancer have resources deployed in multiple AZs, actively handing requests. The architecture also demonstrates hot

standby, where if the primary Amazon RDS instance fails (or the AZ itself fails), then the standby instance is promoted to primary.



Figure 24: Multi-AZ architecture

In addition to this HA architecture, you need to add backups of all data required to run your workload. This is especially important for data that is constrained to a single zone such as Amazon EBS volumes or Amazon Redshift clusters. If an AZ fails, you will need to restore this data to another AZ. Where possible, you should also copy data backups to another AWS Region as an additional layer of protection.

An less common alternative approach to single Region, multi-AZ DR is illustrated in the blog post, Building highly resilient applications using Amazon Application Recovery Controller, Part 1: Single-Region stack. Here, the strategy is to maintain as much isolation between the AZs as possible, like how Regions operate. Using this alternative strategy, you can choose an active/active or active/passive approach.

> **Note**
>
> Some workloads have regulatory data residency requirements. If this applies to your workload in a locality that currently has only one AWS Region, then multi-Region will not suit your business needs. Multi-AZ strategies provide good protection against most disasters.

3. **Assess the resources of your workload, and what their configuration will be in the recovery Region prior to failover (during normal operation).**

   For infrastructure and AWS resources use infrastructure as code such as AWS CloudFormation or third-party tools like Hashicorp Terraform. To deploy across multiple accounts and Regions with a single operation you can use AWS CloudFormation StackSets. For Multi-site active/active and Hot Standby strategies, the deployed infrastructure in your recovery Region has the same resources as your primary Region. For Pilot Light and Warm Standby strategies, the deployed infrastructure will require additional actions to become production ready. Using CloudFormation parameters and conditional logic, you can control whether a deployed stack is active or standby with a single template. When using Elastic Disaster Recovery, the service will replicate and orchestrate the restoration of application configurations and compute resources.

   All DR strategies require that data sources are backed up within the AWS Region, and then those backups are copied to the recovery Region. AWS Backup provides a centralized view where you can configure, schedule, and monitor backups for these resources. For Pilot Light, Warm Standby, and Multi-site active/active, you should also replicate data from the primary Region to data resources in the recovery Region, such as Amazon Relational Database Service (Amazon RDS) DB instances or Amazon DynamoDB tables. These data resources are therefore live and ready to serve requests in the recovery Region.

   To learn more about how AWS services operate across Regions, see this blog series on Creating a Multi-Region Application with AWS Services.

4. **Determine and implement how you will make your recovery Region ready for failover when needed (during a disaster event).**

   For multi-site active/active, failover means evacuating a Region, and relying on the remaining active Regions. In general, those Regions are ready to accept traffic. For Pilot Light and Warm Standby strategies, your recovery actions will need to deploy the missing resources, such as the EC2 instances in Figure 20, plus any other missing resources.

   For all of the above strategies you may need to promote read-only instances of databases to become the primary read/write instance.

   For backup and restore, restoring data from backup creates resources for that data such as EBS volumes, RDS DB instances, and DynamoDB tables. You also need to restore the infrastructure and deploy code. You can use AWS Backup to restore data in the recovery Region. See REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from

sources for more details. Rebuilding the infrastructure includes creating resources like EC2 instances in addition to the Amazon Virtual Private Cloud (Amazon VPC), subnets, and security groups needed. You can automate much of the restoration process. To learn how, see this blog post.

5. **Determine and implement how you will reroute traffic to failover when needed (during a disaster event).**

   This failover operation can be initiated either automatically or manually. Automatically initiated failover based on health checks or alarms should be used with caution since an unnecessary failover (false alarm) incurs costs such as non-availability and data loss. Manually initiated failover is therefore often used. In this case, you should still automate the steps for failover, so that the manual initiation is like the push of a button.

   There are several traffic management options to consider when using AWS services. One option is to use Amazon Route 53. Using Amazon Route 53, you can associate multiple IP endpoints in one or more AWS Regions with a Route 53 domain name. To implement manually initiated failover you can use Amazon Application Recovery Controller, which provides a highly available data plane API to reroute traffic to the recovery Region. When implementing failover, use data plane operations and avoid control plane ones as described in REL11-BP04 Rely on the data plane and not the control plane during recovery.

   To learn more about this and other options see this section of the Disaster Recovery Whitepaper.

6. **Design a plan for how your workload will fail back.**

   Failback is when you return workload operation to the primary Region, after a disaster event has abated. Provisioning infrastructure and code to the primary Region generally follows the same steps as were initially used, relying on infrastructure as code and code deployment pipelines. The challenge with failback is restoring data stores, and ensuring their consistency with the recovery Region in operation.

   In the failed over state, the databases in the recovery Region are live and have the up-to-date data. The goal then is to re-synchronize from the recovery Region to the primary Region, ensuring it is up-to-date.

   Some AWS services will do this automatically. If using Amazon DynamoDB global tables, even if the table in the primary Region had become not available, when it comes back online, DynamoDB resumes propagating any pending writes. If using Amazon Aurora Global Database and using managed planned failover, then Aurora global database's existing replication topology

is maintained. Therefore, the former read/write instance in the primary Region will become a replica and receive updates from the recovery Region.

In cases where this is not automatic, you will need to re-establish the database in the primary Region as a replica of the database in the recovery Region. In many cases this will involve deleting the old primary database, and creating new replicas.

After a failover, if you can continue running in your recovery Region, consider making this the new primary Region. You would still do all the above steps to make the former primary Region into a recovery Region. Some organizations do a scheduled rotation, swapping their primary and recovery Regions periodically (for example every three months).

All of the steps required to fail over and fail back should be maintained in a playbook that is available to all members of the team, and is periodically reviewed.

When using Elastic Disaster Recovery, the service will assist in orchestrating and automating the failback process. For more details, see Performing a failback.

Level of effort for the Implementation Plan: High

Resources

Related best practices:

- the section called “REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources”
- the section called “REL11-BP04 Rely on the data plane and not the control plane during recovery”
- the section called “REL13-BP01 Define recovery objectives for downtime and data loss”

Related documents:

- AWS Architecture Blog: Disaster Recovery Series
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- Disaster recovery options in the cloud
- Build a serverless multi-region, active-active backend solution in an hour
- Multi-region serverless backend — reloaded

- RDS: Replicating a Read Replica Across Regions
- Route 53: Configuring DNS Failover
- S3: Cross-Region Replication
- What Is AWS Backup?
- What is Amazon Application Recovery Controller?
- AWS Elastic Disaster Recovery
- HashiCorp Terraform: Get Started - AWS
- APN Partner: partners that can help with disaster recovery
- AWS Marketplace: products that can be used for disaster recovery

Related videos:

- Disaster Recovery of Workloads on AWS
- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- Get Started with AWS Elastic Disaster Recovery | Amazon Web Services

REL13-BP03 Test disaster recovery implementation to validate the implementation

Regularly test failover to your recovery site to verify that it operates properly and that RTO and RPO are met.

Common anti-patterns:

- Never exercise failovers in production.

Benefits of establishing this best practice: Regularly testing you disaster recovery plan verifies that it will work when it needs to, and that your team knows how to perform the strategy.

Level of risk exposed if this best practice is not established: High

Implementation guidance

A pattern to avoid is developing recovery paths that are rarely exercised. For example, you might have a secondary data store that is used for read-only queries. When you write to a data store and the primary fails, you might want to fail over to the secondary data store. If you don't frequently

test this failover, you might find that your assumptions about the capabilities of the secondary data store are incorrect. The capacity of the secondary, which might have been sufficient when you last tested, might be no longer be able to tolerate the load under this scenario. Our experience has shown that the only error recovery that works is the path you test frequently. This is why having a small number of recovery paths is best. You can establish recovery patterns and regularly test them. If you have a complex or critical recovery path, you still need to regularly exercise that failure in production to convince yourself that the recovery path works. In the example we just discussed, you should fail over to the standby regularly, regardless of need.

Implementation steps

1. Engineer your workloads for recovery. Regularly test your recovery paths. Recovery-oriented computing identifies the characteristics in systems that enhance recovery: isolation and redundancy, system-wide ability to roll back changes, ability to monitor and determine health, ability to provide diagnostics, automated recovery, modular design, and ability to restart. Exercise the recovery path to verify that you can accomplish the recovery in the specified time to the specified state. Use your runbooks during this recovery to document problems and find solutions for them before the next test.
2. For Amazon EC2-based workloads, use AWS Elastic Disaster Recovery to implement and launch drill instances for your DR strategy. AWS Elastic Disaster Recovery provides the ability to efficiently run drills, which helps you prepare for a failover event. You can also frequently launch of your instances using Elastic Disaster Recovery for test and drill purposes without redirecting the traffic.

Resources

Related documents:

- APN Partner: partners that can help with disaster recovery
- AWS Architecture Blog: Disaster Recovery Series
- AWS Marketplace: products that can be used for disaster recovery
- AWS Elastic Disaster Recovery
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- AWS Elastic Disaster Recovery Preparing for Failover
- The Berkeley/Stanford recovery-oriented computing project
- What is AWS Fault Injection Simulator?

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications
- AWS re:Invent 2019: Backup-and-restore and disaster-recovery solutions with AWS

REL13-BP04 Manage configuration drift at the DR site or Region

To perform a successful disaster recovery (DR) procedure, your workload must be able to resume normal operations in a timely manner with no relevant loss of functionality or data once the DR environment has been brought online. To achieve this goal, it's essential to maintain consistent infrastructure, data, and configurations between your DR environment and the primary environment.

Desired outcome: Your disaster recovery site's configuration and data are in parity with the primary site, which facilitates rapid and complete recovery when needed.

Common anti-patterns:

- You fail to update recovery locations when changes are made to the primary locations, which results in outdated configurations that could hinder recovery efforts.
- You do not consider potential limitations such as service differences between primary and recovery locations, which can lead to unexpected failures during failover.
- You rely on manual processes to update and synchronize the DR environment, which increases the risk of human error and inconsistency.
- You fail to detect configuration drift, which leads to a false sense of DR site readiness prior to an incident.

Benefits of establishing this best practice: Consistency between the DR environment and the primary environment significantly improves the likelihood of a successful recovery after an incident and reduces the risk of a failed recovery procedure.

Level of risk exposed if this best practice is not established: High

Implementation guidance

A comprehensive approach to configuration management and failover readiness can help you verify that the DR site is consistently updated and prepared to take over in the event of a primary site failure.

To achieve consistency between your primary and disaster recovery (DR) environments, validate that your delivery pipelines distribute applications to both your primary and DR sites. Roll out changes to the DR sites after an appropriate evaluation period (also known as *staggered deployments*) to detect problems at the primary site and halt the deployment before they spread. Implement monitoring to detect configuration drift, and track changes and compliance across your environments. Perform automated remediation in the DR site to keep it fully consistent and ready to take over in the event of an incident.

Implementation steps

1. Validate that the DR region contains the AWS services and features required for a successful execution of your DR plan.
2. Use infrastructure as code (IaC). Keep your production infrastructure and application configuration templates accurate, and regularly apply them to your disaster recovery environment. AWS CloudFormation can detect drift between what your CloudFormation templates specify and what is actually deployed.
3. Configure CI/CD pipelines to deploy applications and infrastructure updates to all environments, including primary and DR sites. CI/CD solutions such as AWS CodePipeline can automate the deployment process, which reduces the risk of configuration drift.
4. Stagger deployments between the primary and DR environments. This approach allows updates to be initially deployed and tested in the primary environment, which isolates issues in the primary site before they are propagated to the DR site. This approach prevents defects from being simultaneously pushed to production and the DR site at the same time and maintains the integrity of the DR environment.
5. Continually monitor resource configurations in both primary and DR environments. Solutions such as AWS Config can help to enforce configuration compliance and detect drift, which helps maintain the consistent configurations across environments.
6. Implement alerting mechanisms to track and notify upon any configuration drift or data replication interruption or lag.
7. Automate the remediation of detected configuration drift.
8. Schedule regular audits and compliance checks to verify ongoing alignment between primary and DR configurations. Periodic reviews help you maintain compliance with defined rules and identify any discrepancies that need to be addressed.
9. Check for mismatches in AWS provisioned capacity, service quotas, throttle limits, and configuration and version discrepancies.

Resources

Related best practices:

- REL01-BP01 Aware of service quotas and constraints
- REL01-BP02 Manage service quotas across accounts and regions
- REL01-BP04 Monitor and manage quotas
- REL13-BP03 Test disaster recovery implementation to validate the implementation

Related documents:

- Remediating Noncompliant AWS Resources by AWS Config Rules
- AWS Systems Manager Automation
- AWS CloudFormation: Detecting unmanaged configuration changes to stacks and resources
- AWS CloudFormation: Detect Drift on an Entire CloudFormation Stack
- AWS Systems Manager Automation
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- How do I implement an Infrastructure Configuration Management solution on AWS?
- Remediating Noncompliant AWS Resources by AWS Config Rules

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)

Related examples:

- CloudFormation Registry
- Quota Monitor for AWS
- Implement automatic drift remediation for AWS CloudFormation using Amazon CloudWatch and AWS Lambda
- AWS Architecture Blog: Disaster Recovery Series
- AWS Marketplace: products that can be used for disaster recovery
- Automating safe, hands-off deployments

REL13-BP05 Automate recovery

Implement tested and automated recovery mechanisms that are reliable, observable, and reproducible to reduce the risk and business impact of failure.

Desired outcome: You have implemented a well-documented, standardized, and thoroughly-tested automation workflow for recovery processes. Your recovery automation automatically corrects minor issues that pose low risk of data loss or unavailability. You are able to quickly invoke recovery processes for serious incidents, observe the remediation behavior while they operate, and end the processes if you observe dangerous situations or failures.

Common anti-patterns:

- You depend on components or mechanisms that are in a failed or degraded state as part of your recovery plan.
- Your recovery processes require manual intervention, such as console access (also known as *click ops*).
- You automatically initiate recovery procedures in situations that present a high risk of data loss or unavailability.
- You fail to include a mechanism to abort a recovery procedure (like an *Andon cord* or *big red stop button*) that is not working or that poses additional risks.

Benefits of establishing this best practice:

- Increased reliability, predictability, and consistency of recovery operations.
- Ability to meet more stringent recovery objectives, including Recovery Time Objective (RTO) and Recovery Point Objective (RPO).
- Reduced likelihood of recovery failing during an incident.
- Reduced risk of failures associated with manual recovery processes that are prone to human error.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To implement automated recovery, you need a comprehensive approach that uses AWS services and best practices. To start, identify critical components and potential failure points in your

workload. Develop automated processes that can recover your workloads and data from failures without human intervention.

Develop your recovery automation using infrastructure as code (IaC) principles. This makes your recovery environment consistent with the source environment and allows for version control of your recovery processes. To orchestrate complex recovery workflows, consider solutions such as AWS Systems Manager Automations or AWS Step Functions.

Automation of recovery processes provides significant benefits and can help you more easily achieve your Recovery Time Objective (RTO) and Recovery Point Objective (RPO). However, they can encounter unexpected situations that may cause them to fail or create new risks of their own such as additional downtime and data loss. To mitigate this risk, provide the ability to quickly halt a recovery automation in progress. Once halted, you can investigate and take corrective steps.

For supported workloads, consider solutions such as AWS Elastic Disaster Recovery (AWS DRS) to provide automated failover. AWS DRS continually replicates your machines (including operating system, system state configuration, databases, applications, and files) into a staging area in your target AWS account and preferred Region. If an incident occurs, AWS DRS automates the conversion of your replicated servers into fully-provisioned workloads in your recovery Region on AWS.

Maintenance and improvement of automated recovery is an ongoing process. Continually test and refine your recovery procedures based on lessons learned, and stay updated on new AWS services and features that can enhance your recovery capabilities.

Implementation steps

1. **Plan for automated recovery**
    a. Conduct a thorough review of your workload architecture, components, and dependencies to identify and plan automated recovery mechanisms. Categorize your workload's dependencies into *hard* and *soft* dependencies. Hard dependencies are those that the workload cannot operate without and for which no substitute can be provided. Soft dependencies are those that the workload ordinarily uses but are replaceable with temporary substitute systems or processes or can be handled by graceful degradation.
    b. Establish processes to identify and recover missing or corrupted data.
    c. Define steps to confirm a recovered steady state after recovery actions have been completed.
    d. Consider any actions required to make the recovered system ready for full service, such as pre-warming and populating caches.

e. Consider problems that could be encountered during the recovery process and how to detect and remediate them.

f. Consider scenarios where the primary site and its control plane are inaccessible. Verify that recovery actions can be performed independently without reliance on the primary site. Consider solutions such as Amazon Application Recovery Controller (ARC) to redirect traffic without the need to manually mutate DNS records.

2. **Develop automated recovery process**

a. Implement automated fault detection and failover mechanisms for hands-free recovery. Build dashboards such as with Amazon CloudWatch to report the progress and health of automated recovery procedures. Include procedures to validate successful recovery. Provide a mechanism to abort a recovery in process.

b. Build playbooks as a fallback process for faults that cannot be automatically recovered from, and take into consideration your disaster recovery plan.

c. Test recovery processes as discussed in REL13-BP03.

3. **Prepare for recovery**

a. Evaluate the state of your recovery site and deploy critical components to it in advance. For more detail, see REL13-BP04.

b. Define clear roles, responsibilities, and decision-making processes for recovery operations, involving relevant stakeholders and teams across the organization.

c. Define the conditions to initiate your recovery processes.

d. Create a plan to revert the recovery process and fall back to your primary site if required or after it's considered safe.

Resources

Related best practices:

- REL07-BP01 Use automation when obtaining or scaling resources
- REL11-BP01 Monitor all components of the workload to detect failures
- REL13-BP02 Use defined recovery strategies to meet the recovery objectives
- REL13-BP03 Test disaster recovery implementation to validate the implementation
- REL13-BP04 Manage configuration drift at the DR site or Region

Related documents:

- AWS Architecture Blog: Disaster Recovery Series
- Disaster Recovery of Workloads on AWS: Recovery in the Cloud (AWS Whitepaper)
- Orchestrate Disaster Recovery Automation using Amazon Route 53 ARC and AWS Step Functions
- Build AWS Systems Manager Automation runbooks using AWS CDK
- AWS Marketplace: Products That Can Be Used for Disaster Recovery
- AWS Systems Manager Automation
- AWS Elastic Disaster Recovery
- Using Elastic Disaster Recovery for Failover and Failback
- AWS Elastic Disaster Recovery Resources
- APN Partner: Partners That Can Help with Disaster Recovery

Related videos:

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications (ARC209-R2)
- AWS re:Invent 2022: AWS On Air ft. AWS Failback for AWS Elastic Disaster Recovery

Performance efficiency

The Performance Efficiency pillar includes the ability to use cloud resources efficiently to meet performance requirements, and to maintain that efficiency as demand changes and technologies evolve. You can find prescriptive guidance on implementation in the Performance Efficiency Pillar whitepaper.

Best practice areas

- Architecture selection
- Compute and hardware
- Data management
- Networking and content delivery
- Process and culture

Architecture selection

Questions

- PERF 1. How do you select appropriate cloud resources and architecture for your workload?

PERF 1. How do you select appropriate cloud resources and architecture for your workload?

The optimal solution for a particular workload varies, and solutions often combine multiple approaches. Well-Architected workloads use multiple solutions and allow different features to improve performance.

Best practices

- PERF01-BP01 Learn about and understand available cloud services and features
- PERF01-BP02 Use guidance from your cloud provider or an appropriate partner to learn about architecture patterns and best practices
- PERF01-BP03 Factor cost into architectural decisions
- PERF01-BP04 Evaluate how trade-offs impact customers and architecture efficiency
- PERF01-BP05 Use policies and reference architectures
- PERF01-BP06 Use benchmarking to drive architectural decisions
- PERF01-BP07 Use a data-driven approach for architectural choices

PERF01-BP01 Learn about and understand available cloud services and features

Continually learn about and discover available services and configurations that help you make better architectural decisions and improve performance efficiency in your workload architecture.

Common anti-patterns:

- You use the cloud as a collocated data center.
- You do not modernize your application after migration to the cloud.
- You only use one storage type for all things that need to be persisted.
- You use instance types that are closest matched to your current standards, but are larger where needed.

- You deploy and manage technologies that are available as managed services.

Benefits of establishing this best practice: By considering new services and configurations, you may be able to greatly improve performance, reduce cost, and optimize the effort required to maintain your workload. It can also help you accelerate the time-to-value for cloud-enabled products.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS continually releases new services and features that can improve performance and reduce the cost of cloud workloads. Staying up-to-date with these new services and features is crucial for maintaining performance efficacy in the cloud. Modernizing your workload architecture also helps you accelerate productivity, drive innovation, and unlock more growth opportunities.

Implementation steps

- Inventory your workload software and architecture for related services. Decide which category of products to learn more about.
- Explore AWS offerings to identify and learn about the relevant services and configuration options that can help you improve performance and reduce cost and operational complexity.
  - Amazon Web Services Cloud
  - AWS Academy
  - What's New with AWS?
  - AWS Blog
  - AWS Skill Builder
  - AWS Events and Webinars
  - AWS Training and Certifications
  - AWS Youtube Channel
  - AWS Workshops
  - AWS Communities
- Use Amazon Q to get relevant information and advice about services.
- Use sandbox (non-production) environments to learn and experiment with new services without incurring extra cost.
- Continually learn about new cloud services and features.

Resources

Related documents:

- Overview of Amazon Web Services
- Amazon EC2 features
- Learn step-by-step with an AWS Partner Learning Plan
- AWS Training and Certification
- My learning path to become an AWS solutions architect
- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Build modern applications on AWS

Related videos:

- AWS re:Invent 2023 - What's new with Amazon EC2
- AWS re:Invent 2022 - Reduce your operational and infrastructure costs with Amazon ECS
- AWS re:Invent 2023 - Build with the efficiency, agility & innovation of the cloud with AWS
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- This is my Architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP02 Use guidance from your cloud provider or an appropriate partner to learn about architecture patterns and best practices

Use cloud company resources such as documentation, solutions architects, professional services, or appropriate partners to guide your architectural decisions. These resources help you review and improve your architecture for optimal performance.

Common anti-patterns:

- You use AWS as a common cloud provider.
- You use AWS services in a manner that they were not designed for.
- You follow all guidance without considering your business context.

Benefits of establishing this best practice: Using guidance from a cloud provider or an appropriate partner can help you to make the right architectural choices for your workload and give you confidence in your decisions.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS offers a wide range of guidance, documentation, and resources that can help you build and manage efficient cloud workloads. AWS documentation provides code samples, tutorials, and detailed service explanations. In addition to documentation, AWS provides training and certification programs, solutions architects, and professional services that can help customers explore different aspects of cloud services and implement efficient cloud architecture on AWS.

Leverage these resources to gain insights into valuable knowledge and best practices, save time, and achieve better outcomes in the AWS Cloud.

Implementation steps

- Review AWS documentation and guidance and follow the best practices. These resources can help you effectively choose and configure services and achieve better performance.
  - AWS documentation (like user guides and whitepapers)
  - AWS Blog
  - AWS Training and Certifications
  - AWS Youtube Channel
- Join AWS partner events (like AWS Global Summits, AWS re:Invent, user groups, and workshops) to learn from AWS experts about best practices for using AWS services.
  - Learn step-by-step with an AWS Partner Learning Plan
  - AWS Events and Webinars
  - AWS Workshops

  - AWS Communities
- Reach out to AWS for assistance when you need additional guidance or product information. AWS Solutions Architects and AWS Professional Services provide guidance for solution implementation. AWS Partners provide AWS expertise to help you unlock agility and innovation for your business.
- Use Support if you need technical support to use a service effectively. Our Support plans are designed to give you the right mix of tools and access to expertise so that you can be successful with AWS while optimizing performance, managing risk, and keeping costs under control.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- AWS Enterprise Support

Related videos:

- This is my Architecture
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Implementing distributed design patterns on AWS
- AWS re:Invent 2023 - Application architecture as code

Related examples:

- AWS Samples
- AWS SDK Examples
- AWS Analytics Reference Architecture

PERF01-BP03 Factor cost into architectural decisions

Factor cost into your architectural decisions to improve resource utilization and performance efficiency of your cloud workload. When you are aware of the cost implications of your cloud workload, you are more likely to leverage efficient resources and reduce wasteful practices.

Common anti-patterns:

- You only use one family of instances.
- You do not evaluate licensed solutions against open-source solutions.
- You do not define storage lifecycle policies.
- You do not review new services and features of the AWS Cloud.
- You only use block storage.

Benefits of establishing this best practice: Factoring cost into your decision making allows you to use more efficient resources and explore other investments.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Optimizing workloads for cost can improve resource utilization and avoid waste in a cloud workload. Factoring cost into architectural decisions usually includes right-sizing workload components and enabling elasticity, which results in improved cloud workload performance efficiency.

Implementation steps

- Establish cost objectives like budget limits for your cloud workload.
- Identify the key components (like instances and storage) that drive cost of your workload. You can use AWS Pricing Calculator and AWS Cost Explorer to identify key cost drivers in your workload.
- Understand pricing models in the cloud, such as On-Demand, Reserved Instances, Savings Plans, and Spot Instances.
- Use Well-Architected cost optimization best practices to optimize these key components for cost.
- Continually monitor and analyze cost to identify cost optimization opportunities in your workload.

- Use AWS Budgets to get alerts for unacceptable costs.
- Use AWS Compute Optimizer or AWS Trusted Advisor to get cost optimization recommendations.
- Use AWS Cost Anomaly Detection to get automated cost anomaly detection and root cause analysis.

Resources

Related documents:

- What is AWS Billing and Cost Management?
- Cost Optimization with AWS
- Choosing an AWS cost management strategy
- A Beginner's Guide to AWS Cost Management
- A Detailed Overview of the Cost Intelligence Dashboard
- AWS Architecture Center
- AWS Solutions Library
- AWS Knowledge Center

Related videos:

- This is my Architecture
- AWS re:Invent 2023 - What's new with AWS cost optimization
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

Related examples:

- AWS Compute Optimizer Demo code
- Cost Optimization Workshop
- Cloud Financial Management Technical Implementation Playbooks
- Startup optimization: Tuning application performance for maximum efficiency

- Serverless Optimization Workshop (Performance and Cost)
- Scaling cost effective architectures

PERF01-BP04 Evaluate how trade-offs impact customers and architecture efficiency

When evaluating performance-related improvements, determine which choices impact your customers and workload efficiency. For example, if using a key-value data store increases system performance, it is important to evaluate how the eventually consistent nature of this change will impact customers.

Common anti-patterns:

- You assume that all performance gains should be implemented, even if there are tradeoffs for implementation.
- You only evaluate changes to workloads when a performance issue has reached a critical point.

Benefits of establishing this best practice: When you are evaluating potential performance-related improvements, you must decide if the tradeoffs for the changes are acceptable with the workload requirements. In some cases, you may have to implement additional controls to compensate for the tradeoffs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Identify critical areas in your architecture in terms of performance and customer impact. Determine how you can make improvements, what trade-offs those improvements bring, and how they impact the system and the user experience. For example, implementing caching data can help dramatically improve performance but requires a clear strategy for how and when to update or invalidate cached data to prevent incorrect system behavior.

Implementation steps

- Understand your workload requirements and SLAs.
- Clearly define evaluation factors. Factors may relate to cost, reliability, security, and performance of your workload.
- Select architecture and services that can address your requirements.

- Conduct experimentation and proof of concepts (POCs) to evaluate trade-off factors and impact on customers and architecture efficiency. Usually, highly-available, performant, and secure workloads consume more cloud resources while providing better customer experience. Understand the trade-offs across your workload's complexity, performance, and cost. Typically, prioritizing two of the factors comes at the expense of the third.

Resources

Related documents:

- Amazon Builders' Library
- Quick KPIs
- Amazon CloudWatch RUM
- X-Ray Documentation
- Understand resiliency patterns and trade-offs to architect efficiently in the cloud

Related videos:

- Optimize applications through Amazon CloudWatch RUM
- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP05 Use policies and reference architectures

Use internal policies and existing reference architectures when selecting services and configurations to be more efficient when designing and implementing your workload.

Common anti-patterns:

- You allow a wide variety of technology that may impact the management overhead of your company.

Benefits of establishing this best practice: Establishing a policy for architecture, technology, and vendor choices allows decisions to be made quickly.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Having internal policies in selecting resources and architecture provides standards and guidelines to follow when making architectural choices. Those guidelines streamline the decision-making process when choosing the right cloud service and can help improve performance efficiency. Deploy your workload using policies or reference architectures. Integrate the services into your cloud deployment, then use your performance tests to verify that you can continue to meet your performance requirements.

Implementation steps

- Clearly understand the requirements of your cloud workload.
- Review internal and external policies to identify the most relevant ones.
- Use the appropriate reference architectures provided by AWS or your industry best practices.
- Create a continuum consisting of policies, standards, reference architectures, and prescriptive guidelines for common situations. Doing so allows your teams to move faster. Tailor the assets for your vertical if applicable.
- Validate these policies and reference architectures for your workload in sandbox environments.
- Stay up-to-date with industry standards and AWS updates to make sure your policies and reference architectures help optimize your cloud workload.

Resources

Related documents:

- AWS Architecture Center
- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- AWS Architecture Blog

Related videos:

- This is my Architecture
- AWS re:Invent 2022 - Accelerate value for your business with SAP & AWS reference architecture

Related examples:

- AWS Samples
- AWS SDK Examples

PERF01-BP06 Use benchmarking to drive architectural decisions

Benchmark the performance of an existing workload to understand how it performs on the cloud and drive architectural decisions based on that data.

Common anti-patterns:

- You rely on common benchmarks that are not indicative of your workload's characteristics.
- You rely on customer feedback and perceptions as your only benchmark.

Benefits of establishing this best practice: Benchmarking your current implementation allows you to measure performance improvements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use benchmarking with synthetic tests to assess how your workload's components perform. Benchmarking is generally quicker to set up than load testing and is used to evaluate the technology for a particular component. Benchmarking is often used at the start of a new project, when you lack a full solution to load test.

You can either build your own custom benchmark tests or use an industry standard test, such as TPC-DS, to benchmark your workloads. Industry benchmarks are helpful when comparing environments. Custom benchmarks are useful for targeting specific types of operations that you expect to make in your architecture.

When benchmarking, it is important to pre-warm your test environment to get valid results. Run the same benchmark multiple times to verify that you've captured any variance over time.

Because benchmarks are generally faster to run than load tests, they can be used earlier in the deployment pipeline and provide faster feedback on performance deviations. When you evaluate a significant change in a component or service, a benchmark can be a quick way to see if you can justify the effort to make the change. Using benchmarking in conjunction with load testing is important because load testing informs you about how your workload performs in production.

Implementation steps

- Plan and define:
  - Define the objectives, baseline, testing scenarios, metrics (like CPU utilization, latency, or throughput), and KPIs for your benchmark.
  - Focus on user requirements in terms of user experience and factors such as response time and accessibility.
  - Identify a benchmarking tool that is suitable for your workload. You can use AWS services like Amazon CloudWatch or a third-party tool that is compatible with your workload.
- Configure and instrument:
  - Set up your environment and configure your resources.
  - Implement monitoring and logging to capture testing results.
- Benchmark and monitor:
  - Perform your benchmark tests and monitor the metrics during the test.
- Analyze and document:
  - Document your benchmarking process and findings.
  - Analyze the results to identify bottlenecks, trends, and areas of improvement.
  - Use test results to make architectural decisions and adjust your workload. This may include changing services or adopting new features.
- Optimize and repeat:
  - Adjust resource configurations and allocations based on your benchmarks.
  - Retest your workload after the adjustment to validate your improvements.
  - Document your learnings, and repeat the process to identify other areas of improvement.

Resources

Related documents:

- AWS Architecture Center

- AWS Partner Network
- AWS Solutions Library
- AWS Knowledge Center
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Genomics workflows, Part 5: automated benchmarking
- Benchmark and optimize endpoint deployment in Amazon SageMaker AI JumpStart

Related videos:

- AWS re:Invent 2023 - Benchmarking AWS Lambda cold starts
- Benchmarking stateful services in the cloud
- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- AWS Samples
- AWS SDK Examples
- Distributed Load Tests
- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP07 Use a data-driven approach for architectural choices

Define a clear, data-driven approach for architectural choices to verify that the right cloud services and configurations are used to meet your specific business needs.

Common anti-patterns:

- You assume your current architecture is static and should not be updated over time.
- Your architectural choices are based upon guesses and assumptions.

- You introduce architecture changes over time without justification.

Benefits of establishing this best practice: By having a well-defined approach for making architectural choices, you use data to influence your workload design and make informed decisions over time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use internal experience and knowledge of the cloud or external resources such as published use cases or whitepapers to choose resources and services in your architecture. You should have a well-defined process that encourages experimentation and benchmarking with the services that could be used in your workload.

Backlogs for critical workloads should consist of not just user stories which deliver functionality relevant to business and users, but also technical stories which form an architecture runway for the workload. This runway is informed by new advancements in technology and new services and adopts them based on data and proper justification. This verifies that the architecture remains future-proof and does not stagnate.

Implementation steps

- Engage with key stakeholders to define workload requirements, including performance, availability, and cost considerations. Consider factors such as the number of users and usage pattern for your workload.
- Create an architecture runway or a technology backlog which is prioritized along with the functional backlog.
- Evaluate and assess different cloud services (for more detail, see PERF01-BP01 Learn about and understand available cloud services and features).
- Explore different architectural patterns, like microservices or serverless, that meet your performance requirements (for more detail, see PERF01-BP02 Use guidance from your cloud provider or an appropriate partner to learn about architecture patterns and best practices).
- Consult other teams, architecture diagrams, and resources, such as AWS Solution Architects, AWS Architecture Center, and AWS Partner Network, to help you choose the right architecture for your workload.

- Define performance metrics like throughput and response time that can help you evaluate the performance of your workload.
- Experiment and use defined metrics to validate the performance of the selected architecture.
- Continually monitor and make adjustments as needed to maintain the optimal performance of your architecture.
- Document your selected architecture and decisions as a reference for future updates and learnings.
- Continually review and update the architecture selection approach based on learnings, new technologies, and metrics that indicate a needed change or problem in the current approach.

Resources

Related documents:

- AWS Solutions Library
- AWS Knowledge Center
- Architectural Patterns to Build End-to-End Data Driven Applications on AWS

Related videos:

- This is my Architecture
- AWS re:Invent 2021 - Data-driven enterprise: Going from vision to value
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results

Related examples:

- AWS Samples
- AWS SDK Examples

Compute and hardware

Questions

- PERF 2. How do you select and use compute resources in your workload?

PERF 2. How do you select and use compute resources in your workload?

The optimal compute choice for a particular workload can vary based on application design, usage patterns, and configuration settings. Architectures may use different compute choices for various components and allow different features to improve performance. Selecting the wrong compute choice for an architecture can lead to lower performance efficiency.

Best practices

- PERF02-BP01 Select the best compute options for your workload
- PERF02-BP02 Understand the available compute configuration and features
- PERF02-BP03 Collect compute-related metrics
- PERF02-BP04 Configure and right-size compute resources
- PERF02-BP05 Scale your compute resources dynamically
- PERF02-BP06 Use optimized hardware-based compute accelerators

PERF02-BP01 Select the best compute options for your workload

Selecting the most appropriate compute option for your workload allows you to improve performance, reduce unnecessary infrastructure costs, and lower the operational efforts required to maintain your workload.

Common anti-patterns:

- You use the same compute option that was used on premises.
- You lack awareness of the cloud compute options, features, and solutions, and how those solutions might improve your compute performance.
- You over-provision an existing compute option to meet scaling or performance requirements when an alternative compute option would align to your workload characteristics more precisely.

Benefits of establishing this best practice: By identifying the compute requirements and evaluating against the options available, you can make your workload more resource efficient.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To optimize your cloud workloads for performance efficiency, it is important to select the most appropriate compute options for your use case and performance requirements. AWS provides a variety of compute options that cater to different workloads in the cloud. For instance, you can use Amazon EC2 to launch and manage virtual servers, AWS Lambda to run code without having to provision or manage servers, Amazon ECS or Amazon EKS to run and manage containers, or AWS Batch to process large volumes of data in parallel. Based on your scale and compute needs, you should choose and configure the optimal compute solution for your situation. You can also consider using multiple types of compute solutions in a single workload, as each one has its own advantages and drawbacks.

The following steps guide you through selecting the right compute options to match your workload characteristics and performance requirements.

Implementation steps

- Understand your workload compute requirements. Key requirements to consider include processing needs, traffic patterns, data access patterns, scaling needs, and latency requirements.
- Learn about different AWS compute services for your workload. For more information, see PERF01-BP01 Learn about and understand available cloud services and features. Here are some key AWS compute options, their characteristics, and common use cases:

AWS service	Key characteristics	Common use cases
Amazon Elastic Compute Cloud (Amazon EC2)	Has dedicated option for hardware, license requirements, large selection of different instance families, processor types and compute accelerators	Lift and shift migrations, monolithic application, hybrid environments, enterprise applications
Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS)	Easy deployment, consistent environments, scalable	Microservices, hybrid environments
AWS Lambda	Serverless compute service that runs code in response	Microservices, event-driven applications

AWS service	Key characteristics	Common use cases
	to events and automatically manages the underlying compute resources.	
AWS Batch	Efficiently and dynamically provisions and scales Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Fargate compute resources, with an option to use On-Demand or Spot Instances based on your job requirements	HPC, train ML models
Amazon Lightsail	Preconfigured Linux and Windows application for running small workloads	Simple web applications, custom website

- Evaluate cost (like hourly charge or data transfer) and management overhead (like patching and scaling) associated to each compute option.
- Perform experiments and benchmarking in a non-production environment to identify which compute option can best address your workload requirements.
- Once you have experimented and identified your new compute solution, plan your migration and validate your performance metrics.
- Use AWS monitoring tools like Amazon CloudWatch and optimization services like AWS Compute Optimizer to continually optimize your compute resources based on real-world usage patterns.

Resources

Related documents:

- Cloud Compute with AWS

- Amazon EC2 Instance Types
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Amazon ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration
- Prescriptive Guidance for Containers
- Prescriptive Guidance for Serverless

Related videos:

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AMS
- AWS re:Invent 2023 - What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Powering next-gen Amazon Elastic Compute Cloud: Deep dive on the Nitro System
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- AWS re:Invent 2019 - Amazon Elastic Compute Cloud foundations
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- Amazon EC2 foundations
- Deploy ML models for inference at high performance and low cost

Related examples:

- Migrating the Web application to containers
- Run a Serverless Hello World
- Amazon EKS Workshop
- Amazon EC2 Workshop
- Efficient and Resilient Workloads with Amazon Elastic Compute Cloud Auto Scaling
- Migrating to AWS Graviton with Container Services

PERF02-BP02 Understand the available compute configuration and features

Understand the available configuration options and features for your compute service to help you provision the right amount of resources and improve performance efficiency.

Common anti-patterns:

- You do not evaluate compute options or available instance families against workload characteristics.
- You over-provision compute resources to meet peak-demand requirements.

Benefits of establishing this best practice: Be familiar with AWS compute features and configurations so that you can use a compute solution optimized to meet your workload characteristics and needs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Each compute solution has unique configurations and features available to support different workload characteristics and requirements. Learn how these options complement your workload, and determine which configuration options are best for your application. Examples of these options include instance family, sizes, features (GPU, I/O), bursting, time-outs, function sizes, container instances, and concurrency. If your workload has been using the same compute option for more than four weeks and you anticipate that the characteristics will remain the same in the future, you can use AWS Compute Optimizer to find out if your current compute option is suitable for the workloads from CPU and memory perspective.

Implementation steps

- Understand workload requirements (like CPU need, memory, and latency).
- Review AWS documentation and best practices to learn about recommended configuration options that can help improve compute performance. Here are some key configuration options to consider:

Configuration option	Examples
Instance type	• Compute-optimized instances are ideal for the workloads that require high higher vCPU to memory ratio. • Memory-optimized instances deliver large amounts of memory to support memory intensive workloads. • Storage-optimized instances are designed for workloads that require high, sequentia l read and write access (IOPS) to local storage.
Pricing model	• On-Demand Instances let you use the compute capacity by the hour or second with no long-term commitment. These instances are good for bursting above performance baseline needs. • Savings Plans offer significant savings over On-Demand Instances in exchange for a commitment to use a specific amount of compute power for a one or three-year period. • Spot Instances let you take advantage of unused instance capacity at a discount for your stateless, fault-tolerant workloads.
Auto Scaling	Use Auto Scaling configuration to match compute resources to traffic patterns.

Configuration option	Examples
Sizing	• Use Compute Optimizer to get a machine-learning powered recommendations on which compute configuration best matches your compute characteristics. • Use AWS Lambda Power Tuning to select the best configuration for your Lambda function.
Hardware-based compute accelerators	• Accelerated computing instances perform functions like graphics processing or data pattern matching more efficiently than CPU-based alternatives. • For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload, such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1

Resources

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Processor State Control for Your Amazon EC2 Instance
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Amazon ECS Containers: Amazon ECS Container Instances
- Functions: Lambda Function Configuration

Related videos:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console

- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations
- AWS re:Invent 2022 – Optimizing Amazon EKS for performance and cost on AWS

Related examples:

- Compute Optimizer demo code
- Amazon EC2 spot instances workshop
- Efficient and Resilient Workloads with Amazon EC2 AWS Auto Scaling
- Graviton developer workshop
- AWS for Microsoft workloads immersion day
- AWS for Linux workloads immersion day
- AWS Compute Optimizer Demo code
- Amazon EKS workshop

PERF02-BP03 Collect compute-related metrics

Record and track compute-related metrics to better understand how your compute resources are performing and improve their performance and their utilization.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only use the default metrics recorded by your monitoring software.
- You only review metrics when there is an issue.

Benefits of establishing this best practice: Collecting performance-related metrics will help you align application performance with business requirements to ensure that you are meeting your workload needs. It can also help you continually improve the resource performance and utilization in your workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Cloud workloads can generate large volumes of data such as metrics, logs, and events. In the AWS Cloud, collecting metrics is a crucial step to improve security, cost efficiency, performance, and sustainability. AWS provides a wide range of performance-related metrics using monitoring services such as Amazon CloudWatch to provide you with valuable insights. Metrics such as CPU utilization, memory utilization, disk I/O, and network inbound and outbound can provide insight into utilization levels or performance bottlenecks. Use these metrics as part of a data-driven approach to actively tune and optimize your workload's resources. In an ideal case, you should collect all metrics related to your compute resources in a single platform with retention policies implemented to support cost and operational goals.

Implementation steps

- Identify which performance-related metrics are relevant to your workload. You should collect metrics around resource utilization and the way your cloud workload is operating (like response time and throughput).
  - Amazon EC2 default metrics
  - Amazon ECS default metrics
  - Amazon EKS default metrics
  - Lambda default metrics
  - Amazon EC2 memory and disk metrics
- Choose and set up the right logging and monitoring solution for your workload.
  - AWS native Observability
  - AWS Distro for OpenTelemetry
  - Amazon Managed Service for Prometheus
- Define the required filter and aggregation for the metrics based on your workload requirements.
  - Quantify custom application metrics with Amazon CloudWatch Logs and metric filters
  - Collect custom metrics with Amazon CloudWatch strategic tagging
- Configure data retention policies for your metrics to match your security and operational goals.
  - Default data retention for CloudWatch metrics
  - Default data retention for CloudWatch Logs
- If required, create alarms and notifications for your metrics to help you proactively respond to performance-related issues.

  - Create alarms for custom metrics using Amazon CloudWatch anomaly detection
  - Create metrics and alarms for specific web pages with Amazon CloudWatch RUM
- Use automation to deploy your metric and log aggregation agents.
  - AWS Systems Manager automation
  - OpenTelemetry Collector

Resources

Related documents:

- Monitoring and observability
- Best practices: implementing observability with AWS
- Amazon CloudWatch documentation
- Collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch Agent
- Accessing Amazon CloudWatch Logs for AWS Lambda
- Using CloudWatch Logs with container instances
- Publish custom metrics
- AWS Answers: Centralized Logging
- AWS Services That Publish CloudWatch Metrics
- Monitoring Amazon EKS on AWS Fargate

Related videos:

- AWS re:Invent 2023 – [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 – Implementing application observability
- AWS re:Invent 2023 – Building an effective observability strategy
- AWS re:Invent 2023 – Seamless observability with AWS Distro for OpenTelemetry
- Application Performance Management on AWS

Related examples:

- AWS for Linux Workloads Immersion Day- Amazon CloudWatch

- Monitoring Amazon ECS clusters and containers
- Monitoring with Amazon CloudWatch dashboards
- Amazon EKS workshop

PERF02-BP04 Configure and right-size compute resources

Configure and right-size compute resources to match your workload's performance requirements and avoid under- or over-utilized resources.

Common anti-patterns:

- You ignore your workload performance requirements resulting in over-provisioned or under-provisioned compute resources.
- You only choose the largest or smallest instance available for all workloads.
- You only use one instance family for ease of management.
- You ignore recommendations from AWS Cost Explorer or Compute Optimizer for right-sizing.
- You do not re-evaluate the workload for suitability of new instance types.
- You certify only a small number of instance configurations for your organization.

Benefits of establishing this best practice: Right-sizing compute resources ensures optimal operation in the cloud by avoiding over-provisioning and under-provisioning resources. Properly sizing compute resources typically results in better performance and enhanced customer experience, while also lowering cost.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Right-sizing allows organizations to operate their cloud infrastructure in an efficient and cost-effective manner while addressing their business needs. Over-provisioning cloud resources can lead to extra costs, while under-provisioning can result in poor performance and a negative customer experience. AWS provides tools such as AWS Compute Optimizer and AWS Trusted Advisor that use historical data to provide recommendations to right-size your compute resources.

Implementation steps

- Choose an instance type to best fit your needs:

  - How do I choose the appropriate Amazon EC2 instance type for my workload?
  - Attribute-based instance type selection for Amazon EC2 Fleet
  - Create an Auto Scaling group using attribute-based instance type selection
  - Optimizing your Kubernetes compute costs with Karpenter consolidation
- Analyze the various performance characteristics of your workload and how these characteristics relate to memory, network, and CPU usage. Use this data to choose resources that best match your workload's profile and performance goals.
- Monitor your resource usage using AWS monitoring tools such as Amazon CloudWatch.
- Select the right configuration for compute resources.
  - For ephemeral workloads, evaluate instance Amazon CloudWatch metrics such as `CPUUtilization` to identify if the instance is under-utilized or over-utilized.
  - For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer and AWS Trusted Advisor at regular intervals to identify opportunities to optimize and right-size the compute resource.
- Test configuration changes in a non-production environment before implementing in a live environment.
- Continually re-evaluate new compute offerings and compare against your workload's needs.

Resources

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration
- Processor State Control for Your Amazon EC2 Instance

Related videos:

- Amazon EC2 foundations
- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads

- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

Related examples:

- AWS Compute Optimizer Demo code
- Amazon EKS workshop
- Right-sizing recommendations

PERF02-BP05 Scale your compute resources dynamically

Use the elasticity of the cloud to scale your compute resources up or down dynamically to match your needs and avoid over- or under-provisioning capacity for your workload.

Common anti-patterns:

- You react to alarms by manually increasing capacity.
- You use the same sizing guidelines (generally static infrastructure) as in on-premises.
- You leave increased capacity after a scaling event instead of scaling back down.

Benefits of establishing this best practice: Configuring and testing the elasticity of compute resources can help you save money, maintain performance benchmarks, and improve reliability as traffic changes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS provides the flexibility to scale your resources up or down dynamically through a variety of scaling mechanisms in order to meet changes in demand. Combined with compute-related metrics, a dynamic scaling allows workloads to automatically respond to changes and use the optimal set of compute resources to achieve its goal.

You can use a number of different approaches to match supply of resources with demand.

- **Target-tracking approach**: Monitor your scaling metric and automatically increase or decrease capacity as you need it.
- **Predictive scaling**: Scale in anticipation of daily and weekly trends.
- **Schedule-based approach**: Set your own scaling schedule according to predictable load changes.
- **Service scaling**: Choose services (like serverless) that that automatically scale by design.

You must ensure that workload deployments can handle both scale-up and scale-down events.

Implementation steps

- Compute instances, containers, and functions provide mechanisms for elasticity, either in combination with autoscaling or as a feature of the service. Here are some examples of automatic scaling mechanisms:

Autoscaling Mechanism	Where to use
Amazon EC2 Auto Scaling	To ensure you have the correct number of Amazon EC2 instances available to handle the user load for your application.
Application Auto Scaling	To automatically scale the resources for individual AWS services beyond Amazon EC2 such as AWS Lambda functions or Amazon Elastic Container Service (Amazon ECS) services.
Kubernetes Cluster Autoscaler/Karpenter	To automatically scale Kubernetes clusters.

- Scaling is often discussed related to compute services like Amazon EC2 Instances or AWS Lambda functions. Be sure to also consider the configuration of non-compute services like AWS Glue to match the demand.
- Verify that the metrics for scaling match the characteristics of the workload being deployed. If you are deploying a video transcoding application, 100% CPU utilization is expected and should not be your primary metric. Use the depth of the transcoding job queue instead. You can use a customized metric for your scaling policy if required. To choose the right metrics, consider the following guidance for Amazon EC2:
  - The metric should be a valid utilization metric and describe how busy an instance is.

  - The metric value must increase or decrease proportionally to the number of instances in the Auto Scaling group.
- Make sure that you use dynamic scaling instead of manual scaling for your Auto Scaling group. We also recommend that you use target tracking scaling policies in your dynamic scaling.
- Verify that workload deployments can handle both scaling events (up and down). As an example, you can use Activity history to verify a scaling activity for an Auto Scaling group.
- Evaluate your workload for predictable patterns and proactively scale as you anticipate predicted and planned changes in demand. With predictive scaling, you can eliminate the need to overprovision capacity. For more detail, see Predictive Scaling with Amazon EC2 Auto Scaling.

Resources

Related documents:

- Cloud Compute with AWS
- Amazon EC2 Instance Types
- Amazon ECS Containers: Amazon ECS Container Instances
- Amazon EKS Containers: Amazon EKS Worker Nodes
- Functions: Lambda Function Configuration
- Processor State Control for Your Amazon EC2 Instance
- Deep Dive on Amazon ECS Cluster Auto Scaling
- Introducing Karpenter – An Open-Source High-Performance Kubernetes Cluster Autoscaler

Related videos:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

Related examples:

- Amazon EC2 Auto Scaling Group Examples
- Amazon EKS Workshop
- Scale your Amazon EKS workloads by running on IPv6

PERF02-BP06 Use optimized hardware-based compute accelerators

Use hardware accelerators to perform certain functions more efficiently than CPU-based alternatives.

Common anti-patterns:

- In your workload, you haven't benchmarked a general-purpose instance against a purpose-built instance that can deliver higher performance and lower cost.
- You are using hardware-based compute accelerators for tasks that can be more efficient using CPU-based alternatives.
- You are not monitoring GPU usage.

Benefits of establishing this best practice: By using hardware-based accelerators, such as graphics processing units (GPUs) and field programmable gate arrays (FPGAs), you can perform certain processing functions more efficiently.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Accelerated computing instances provide access to hardware-based compute accelerators such as GPUs and FPGAs. These hardware accelerators perform certain functions like graphic processing or data pattern matching more efficiently than CPU-based alternatives. Many accelerated workloads, such as rendering, transcoding, and machine learning, are highly variable in terms of resource usage. Only run this hardware for the time needed, and decommission them with automation when not required to improve overall performance efficiency.

Implementation steps

- Identify which accelerated computing instances can address your requirements.
- For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload, such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1. AWS Inferentia

instances such as Inf2 instances offer up to 50% better performance/watt over comparable Amazon EC2 instances.

- Collect usage metrics for your accelerated computing instances. For example, you can use CloudWatch agent to collect metrics such as `utilization_gpu` and `utilization_memory` for your GPUs as shown in Collect NVIDIA GPU metrics with Amazon CloudWatch.
- Optimize the code, network operation, and settings of hardware accelerators to make sure that underlying hardware is fully utilized.
  - Optimize GPU settings
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker AI
- Use the latest high performant libraries and GPU drivers.
- Use automation to release GPU instances when not in use.

Resources

Related documents:

- Working with GPUs on Amazon Elastic Container Service
- GPU instances
- Instances with AWS Trainium
- Instances with AWS Inferentia
- Let's Architect! Architecting with custom chips and accelerators

- Accelerated Computing
- Amazon EC2 VT1 Instances
- How do I choose the appropriate Amazon EC2 instance type for my workload?
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

Related videos:

- AWS re:Invent 2021 - How to select Amazon Elastic Compute Cloud GPU instances for deep learning

- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances
- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

Related examples:

- Amazon SageMaker AI and NVIDIA GPU Cloud (NGC)
- Use SageMaker AI with Trainium and Inferentia for optimized deep learning training and inferencing workloads
- Optimizing NLP models with Amazon Elastic Compute Cloud Inf1 instances in Amazon SageMaker AI

Data management

Questions

- PERF 3. How do you store, manage, and access data in your workload?

PERF 3. How do you store, manage, and access data in your workload?

The optimal data management solution for a particular system varies based on the kind of data type (block, file, or object), access patterns (random or sequential), required throughput, frequency of access (online, offline, archival), frequency of update (WORM, dynamic), and availability and durability constraints. Well-Architected workloads use purpose-built data stores which allow different features to improve performance.

Best practices

- PERF03-BP01 Use a purpose-built data store that best supports your data access and storage requirements
- PERF03-BP02 Evaluate available configuration options for data store
- PERF03-BP03 Collect and record data store performance metrics
- PERF03-BP04 Implement strategies to improve query performance in data store
- PERF03-BP05 Implement data access patterns that utilize caching

PERF03-BP01 Use a purpose-built data store that best supports your data access and storage requirements

Understand data characteristics (like shareable, size, cache size, access patterns, latency, throughput, and persistence of data) to select the right purpose-built data stores (storage or database) for your workload.

Common anti-patterns:

- You stick to one data store because there is internal experience and knowledge of one particular type of database solution.
- You assume that all workloads have similar data storage and access requirements.
- You have not implemented a data catalog to inventory your data assets.

Benefits of establishing this best practice: Understanding data characteristics and requirements allows you to determine the most efficient and performant storage technology appropriate for your workload needs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When selecting and implementing data storage, make sure that the querying, scaling, and storage characteristics support the workload data requirements. AWS provides numerous data storage and database technologies including block storage, object storage, streaming storage, file system, relational, key-value, document, in-memory, graph, time series, and ledger databases. Each data management solution has options and configurations available to you to support your use-cases and data models. By understanding data characteristics and requirements, you can break away from monolithic storage technology and restrictive, one-size-fits-all approaches to focus on managing data appropriately.

Implementation steps

- Conduct an inventory of the various data types that exist in your workload.
- Understand and document data characteristics and requirements, including:
  - Data type (unstructured, semi-structured, relational)
  - Data volume and growth
  - Data durability: persistent, ephemeral, transient

- ACID (atomicity, consistency, isolation, durability) requirements
- Data access patterns (read-heavy or write-heavy)
- Latency
- Throughput
- IOPS (input/output operations per second)
- Data retention period

- Learn about different data stores (storage and database services) available for your workload on AWS that can meet your data characteristics, as outlined in PERF01-BP01 Learn about and understand available cloud services and features. Some examples of AWS storage technologies and their key characteristics include:

Type	AWS Services	Key characteristics
Object storage	Amazon S3	Unlimited scalability, high availability, and multiple options for accessibility. Transferring and accessing objects in and out of Amazon S3 can use a service, such as Transfer Acceleration or Access Points, to support your location, security needs, and access patterns.
Archiving storage	Amazon Glacier	Built for data archiving.
Streaming storage	Amazon Kinesis Amazon Managed Streaming for Apache Kafka (Amazon MSK)	Efficient ingestion and storage of streaming data.
Shared file system	Amazon Elastic File System (Amazon EFS)	Mountable file system that can be accessed by multiple types of compute solutions.

Type	AWS Services	Key characteristics
Shared file system	Amazon FSx	Built on the latest AWS compute solutions to support four commonly used file systems: NetApp ONTAP, OpenZFS, Windows File Server, and Lustre. Amazon FSx latency, throughput, and IOPS vary per file system and should be considered when selecting the right file system for your workload needs.
Block storage	Amazon Elastic Block Store (Amazon EBS)	Scalable, high-performance block-storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). Amazon EBS includes SSD-backed storage for transactional, IOPS-intensive workloads and HDD-backe d storage for throughput-intensive workloads.

Type	AWS Services	Key characteristics
Relational database	Amazon Aurora, Amazon RDS, Amazon Redshift.	Designed to support ACID (atomicity, consistency, isolation, durability) transactions, and maintain referential integrity and strong data consistency. Many traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), and ecommerce use relational databases to store their data.
Key-value database	Amazon DynamoDB	Optimized for common access patterns, typically to store and retrieve large volumes of data. High-traffic web apps, ecommerce systems, and gaming applications are typical use-cases for key-value databases.
Document database	Amazon DocumentDB	Designed to store semi-structured data as JSON-like documents. These databases help developers build and update applications such as content management, catalogs, and user profiles quickly.

Type	AWS Services	Key characteristics
In-memory database	Amazon ElastiCache , Amazon MemoryDB for Redis	Used for applications that require real-time access to data, lowest latency and highest throughput. You may use in-memory databases for application caching, session management, gaming leaderboards, low latency ML feature store, microserv ices messaging system, and a high-throughput streaming mechanism
Graph database	Amazon Neptune	Used for applications that must navigate and query millions of relationships between highly connected graph datasets with milliseco nd latency at large scale. Many companies use graph databases for fraud detection, social networkin g, and recommendation engines.
Time Series database	Amazon Timestream	Used to efficiently collect, synthesize, and derive insights from data that changes over time. IoT applications, DevOps, and industrial telemetry can utilize time-series databases.

Type	AWS Services	Key characteristics
Wide column	Amazon Keyspaces (for Apache Cassandra)	Uses tables, rows, and columns, but unlike a relational database, the names and format of the columns can vary from row to row in the same table. You typically see a wide column store in high scale industrial apps for equipment maintenance, fleet management, and route optimization.
Ledger	Amazon Quantum Ledger Database (Amazon QLDB)	Provides a centralized and trusted authority to maintain a scalable, immutable, and cryptographically verifiabl e record of transactions for every application. We see ledger databases used for systems of record, supply chain, registrations, and even banking transactions.

- If you are building a data platform, leverage modern data architecture on AWS to integrate your data lake, data warehouse, and purpose-built data stores.
- The key questions that you need to consider when choosing a data store for your workload are as follows:

Question	Things to consider
How is the data structured?	• If the data is unstructured, consider an object-store such as Amazon S3 or a NoSQL database such as Amazon DocumentDB

Question	Things to consider
	• For key-value data, consider DynamoDB, Amazon ElastiCache (Redis OSS) or Amazon MemoryDB
What level of referential integrity is required?	• For foreign key constraints, relational databases such as Amazon RDS and Aurora can provide this level of integrity. • Typically, within a NoSQL data-model, you would de-normalize your data into a single document or collection of documents to be retrieved in a single request rather than joining across documents or tables.
Is ACID (atomicity, consistency, isolation, durability) compliance required?	• If the ACID properties associated with relational databases are required, consider a relational database such as Amazon RDS and Aurora. • If strong consistency is required for NoSQL database, you can use strongly consistent reads with DynamoDB.
How will the storage requirements change over time? How does this impact scalability?	• Serverless databases such as DynamoDB and Amazon Quantum Ledger Database (Amazon QLDB) will scale dynamically. • Relational databases have upper bounds on provisioned storage, and often must be horizontally partitioned using mechanism s such as sharding once they reach these limits.

Question	Things to consider
What is the proportion of read queries in relation to write queries? Would caching be likely to improve performance?	• Read-heavy workloads can benefit from a caching layer, like ElastiCache or DAX if the database is DynamoDB. • Reads can also be offloaded to read replicas with relational databases such as Amazon RDS.
Does storage and modification (OLTP - Online Transaction Processing) or retrieval and reporting (OLAP - Online Analytical Processing) have a higher priority?	• For high-throughput read as-is transacti onal processing, consider a NoSQL database such as DynamoDB. • For high-throughput and complex read patterns (like join) with consistency use Amazon RDS. • For analytical queries, consider a columnar database such as Amazon Redshift or exporting the data to Amazon S3 and performing analytics using Athena or Amazon Quick.
What level of durability does the data require?	• Aurora automatically replicates your data across three Availability Zones within a Region, meaning your data is highly durable with less chance of data loss. • DynamoDB is automatically replicate d across multiple Availability Zones, providing high availability and data durability. • Amazon S3 provides 11 nines of durabilit y. Many database services, such as Amazon RDS and DynamoDB, support exporting data to Amazon S3 for long-term retention and archival.

Question	Things to consider
Is there a desire to move away from commercial database engines or licensing costs?	• Consider open-source engines such as PostgreSQL and MySQL on Amazon RDS or Aurora. • Leverage AWS Database Migration Service and AWS Schema Conversion Tool to perform migrations from commercial database engines to open-source
What is the operational expectation for the database? Is moving to managed services a primary concern?	• Leveraging Amazon RDS instead of Amazon EC2, and DynamoDB or Amazon DocumentDB instead of self-hosting a NoSQL database can reduce operational overhead.
How is the database currently accessed? Is it only application access, or are there business intelligence (BI) users and other connected off-the-shelf applications?	• If you have dependencies on external tooling then you may have to maintain compatibility with the databases they support. Amazon RDS is fully compatible with the difference engine versions that it supports including Microsoft SQL Server, Oracle, MySQL, and PostgreSQL.

- Perform experiments and benchmarking in a non-production environment to identify which data store can address your workload requirements.

Resources

Related documents:

- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon Glacier: Amazon Glacier Documentation

- Amazon S3: Request Rate and Performance Considerations
- Cloud Storage with AWS
- Amazon EBS I/O Characteristics
- Cloud Databases with AWS
- AWS Database Caching
- DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices
- Choose between Amazon EC2 and Amazon RDS
- Best Practices for Implementing Amazon ElastiCache

Related videos:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimizing storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2022: Building modern data architectures on AWS
- AWS re:Invent 2022: Building data mesh architectures on AWS
- AWS re:Invent 2023: Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023: Advanced data modeling with Amazon DynamoDB
- AWS re:Invent 2022: Modernize apps with purpose-built databases
- Amazon DynamoDB deep dive: Advanced design patterns

Related examples:

- AWS Purpose Built Databases Workshop
- Databases for Developers

- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS
- Amazon S3 Examples
- Optimize Data Pattern using Amazon Redshift Data Sharing
- Database Migrations
- MS SQL Server - AWS Database Migration Service (AWS DMS) Replication Demo
- Database Modernization Hands On Workshop
- Amazon Neptune Samples

PERF03-BP02 Evaluate available configuration options for data store

Understand and evaluate the various features and configuration options available for your data stores to optimize storage space and performance for your workload.

Common anti-patterns:

- You only use one storage type, such as Amazon EBS, for all workloads.
- You use provisioned IOPS for all workloads without real-world testing against all storage tiers.
- You are not aware of the configuration options of your chosen data management solution.
- You rely solely on increasing instance size without looking at other available configuration options.
- You are not testing the scaling characteristics of your data store.

Benefits of establishing this best practice: By exploring and experimenting with the data store configurations, you may be able to reduce the cost of infrastructure, improve performance, and lower the effort required to maintain your workloads.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

A workload could have one or more data stores used based on data storage and access requirements. To optimize your performance efficiency and cost, you must evaluate data access patterns to determine the appropriate data store configurations. While you explore data store

options, take into consideration various aspects such as the storage options, memory, compute, read replica, consistency requirements, connection pooling, and caching options. Experiment with these various configuration options to improve performance efficiency metrics.

Implementation steps

- Understand the current configurations (like instance type, storage size, or database engine version) of your data store.
- Review AWS documentation and best practices to learn about recommended configuration options that can help improve the performance of your data store. Key data store options to consider are the following:

Configuration option	Examples
Offloading reads (like read replicas and caching)	• For DynamoDB tables, you can offload reads using DAX for caching. • You can create an Amazon ElastiCac he (Redis OSS) cluster and configure your application to read from the cache first, falling back to the database if the requested item is not present. • Relational databases such as Amazon RDS and Aurora, and provisioned NoSQL databases such as Neptune and Amazon DocumentDB all support adding read replicas to offload the read portions of the workload. • Serverless databases such as DynamoDB will scale automatically. Ensure that you have enough read capacity units (RCU) provisioned to handle the workload.
Scaling writes (like partition key sharding or introducing a queue)	• For relational databases, you can increase the size of the instance to accommoda te an increased workload or increase the provisioned IOPs to allow for an increased throughput to the underlying storage.

Configuration option	Examples
	• You can also introduce a queue in front of your database rather than writing directly to the database. This pattern allows you to decouple the ingestion from the database and control the flow-rate so the database does not get overwhelmed. • Batching your write requests rather than creating many short-lived transactions can help improve throughput in high-write volume relational databases. • Serverless databases like DynamoDB can scale the write throughput automatically or by adjusting the provisioned write capacity units (WCU) depending on the capacity mode. • You can still run into issues with hot partitions when you reach the throughpu t limits for a given partition key. This can be mitigated by choosing a more evenly distributed partition key or by write-sha rding the partition key.

Configuration option	Examples
Policies to manage the lifecycle of your datasets	• You can use Amazon S3 Lifecycle to manage your objects throughout their lifecycle. If your access patterns are unknown, changing, or unpredictable, you can use Amazon S3 Intelligent-Tiering, which monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers. You can leverage Amazon S3 Storage Lens metrics to identify optimization opportunities and gaps in lifecycle management. • Amazon EFS lifecycle management automatically manages file storage for your file systems.
Connection management and pooling	• Amazon RDS Proxy can be used with Amazon RDS and Aurora to manage connections to the database. • Serverless databases such as DynamoDB do not have connections associated with them, but consider the provisioned capacity and automatic scaling policies to deal with spikes in load.

- Perform experiments and benchmarking in non-production environment to identify which configuration option can address your workload requirements.
- Once you have experimented, plan your migration and validate your performance metrics.
- Use AWS monitoring (like Amazon CloudWatch) and optimization (like Amazon S3 Storage Lens) tools to continuously optimize your data store using real-world usage pattern.

Resources

Related documents:

- Cloud Storage with AWS
- Amazon EBS Volume Types
- Amazon EC2 Storage
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre Performance
- Amazon FSx for Windows File Server Performance
- Amazon Glacier: Amazon Glacier Documentation
- Amazon S3: Request Rate and Performance Considerations
- Amazon EBS I/O Characteristics
- Cloud Databases with AWS
- AWS Database Caching
- DynamoDB Accelerator
- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- Amazon Redshift Spectrum best practices
- Amazon DynamoDB best practices

Related videos:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimize storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2023: What's new with AWS file storage
- AWS re:Invent 2023: Dive deep into Amazon DynamoDB

Related examples:

- AWS Purpose Built Databases Workshop
- Databases for Developers
- AWS Modern Data Architecture Immersion Day

- Amazon EBS Autoscale
- Amazon S3 Examples
- Amazon DynamoDB Examples
- AWS Database migration samples
- Database Modernization Workshop
- Working with parameters on your Amazon RDS for Postgress DB

PERF03-BP03 Collect and record data store performance metrics

Track and record relevant performance metrics for your data store to understand how your data management solutions are performing. These metrics can help you optimize your data store, verify that your workload requirements are met, and provide a clear overview on how the workload performs.

Common anti-patterns:

- You only use manual log file searching for metrics.
- You only publish metrics to internal tools used by your team and don't have a comprehensive picture of your workload.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.
- You only monitor system-level metrics and do not capture data access or usage metrics.

Benefits of establishing this best practice: Establishing a performance baseline helps you understand the normal behavior and requirements of workloads. Abnormal patterns can be identified and debugged faster, improving the performance and reliability of the data store.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To monitor the performance of your data stores, you must record multiple performance metrics over a period of time. This allows you to detect anomalies, as well as measure performance against business metrics to verify you are meeting your workload needs.

Metrics should include both the underlying system that is supporting the data store and the database metrics. The underlying system metrics might include CPU utilization, memory, available

disk storage, disk I/O, cache hit ratio, and network inbound and outbound metrics, while the data store metrics might include transactions per second, top queries, average queries rates, response times, index usage, table locks, query timeouts, and number of connections open. This data is crucial to understand how the workload is performing and how the data management solution is used. Use these metrics as part of a data-driven approach to tune and optimize your workload's resources.

Use tools, libraries, and systems that record performance measurements related to database performance.

Implementation steps

- Identify the key performance metrics for your data store to track.
  - Amazon S3 Metrics and dimensions
  - Monitoring metrics for in an Amazon RDS instance
  - Monitoring DB load with Performance Insights on Amazon RDS
  - Overview of Enhanced Monitoring
  - DynamoDB Metrics and dimensions
  - Monitoring DynamoDB Accelerator
  - Monitoring Amazon MemoryDB with Amazon CloudWatch
  - Which Metrics Should I Monitor?
  - Monitoring Amazon Redshift cluster performance
  - Timestream metrics and dimensions
  - Amazon CloudWatch metrics for Amazon Aurora
  - Logging and monitoring in Amazon Keyspaces (for Apache Cassandra)
  - Monitoring Amazon Neptune Resources
- Use an approved logging and monitoring solution to collect these metrics. Amazon CloudWatch can collect metrics across the resources in your architecture. You can also collect and publish custom metrics to surface business or derived metrics. Use CloudWatch or third-party solutions to set alarms that indicate when thresholds are breached.
- Check if data store monitoring can benefit from a machine learning solution that detects performance anomalies.
  - Amazon DevOps Guru for Amazon RDS provides visibility into performance issues and makes recommendations for corrective actions.

- Configure data retention in your monitoring and logging solution to match your security and operational goals.
  - Default data retention for CloudWatch metrics
  - Default data retention for CloudWatch Logs

Resources

Related documents:

- AWS Database Caching
- Amazon Athena top 10 performance tips
- Amazon Aurora best practices
- DynamoDB Accelerator
- Amazon DynamoDB best practices
- Amazon Redshift Spectrum best practices
- Amazon Redshift performance
- Cloud Databases with AWS
- Amazon RDS Performance Insights

Related videos:

- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Database Performance Monitoring and Tuning with Amazon DevOps Guru for Amazon RDS
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- Best Practices for Monitoring Redis Workloads on Amazon ElastiCache

Related examples:

- AWS Dataset Ingestion Metrics Collection Framework

- Amazon RDS Monitoring Workshop
- AWS Purpose Built Databases Workshop

PERF03-BP04 Implement strategies to improve query performance in data store

Implement strategies to optimize data and improve data query to enable more scalability and efficient performance for your workload.

Common anti-patterns:

- You do not partition data in your data store.
- You store data in only one file format in your data store.
- You do not use indexes in your data store.

Benefits of establishing this best practice: Optimizing data and query performance results in more efficiency, lower cost, and improved user experience.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Data optimization and query tuning are critical aspects of performance efficiency in a data store, as they impact the performance and responsiveness of the entire cloud workload. Unoptimized queries can result in greater resource usage and bottlenecks, which reduce the overall efficiency of a data store.

Data optimization includes several techniques to ensure efficient data storage and access. This also help to improve the query performance in a data store. Key strategies include data partitioning, data compression, and data denormalization, which help data to be optimized for both storage and access.

Implementation steps

- Understand and analyze the critical data queries which are performed in your data store.
- Identify the slow-running queries in your data store and use query plans to understand their current state.
  - Analyzing the query plan in Amazon Redshift
  - Using EXPLAIN and EXPLAIN ANALYZE in Athena

- Implement strategies to improve the query performance. Some of the key strategies include:
  - Using a columnar file format (like Parquet or ORC).
  - Compressing data in the data store to reduce storage space and I/O operation.
  - Data partitioning to split data into smaller parts and reduce data scanning time.
    - Partitioning data in Athena
    - Partitions and data distribution
  - Data indexing on the common columns in the query.
  - Use materialized views for frequent queries.
    - Understanding materialized views
    - Creating materialized views in Amazon Redshift
  - Choose the right join operation for the query. When you join two tables, specify the larger table on the left side of join and the smaller table on the right side of the join.
  - Distributed caching solution to improve latency and reduce the number of database I/O operation.
  - Regular maintenance such as vacuuming, reindexing, and running statistics.
- Experiment and test strategies in a non-production environment.

Resources

Related documents:

- Amazon Aurora best practices
- Amazon Redshift performance
- Amazon Athena top 10 performance tips
- AWS Database Caching
- Best Practices for Implementing Amazon ElastiCache
- Partitioning data in Athena

Related videos:

- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Optimize Amazon Athena Queries with New Query Analysis Tools

Related examples:

- AWS Purpose Built Databases Workshop

PERF03-BP05 Implement data access patterns that utilize caching

Implement access patterns that can benefit from caching data for fast retrieval of frequently accessed data.

Common anti-patterns:

- You cache data that changes frequently.
- You rely on cached data as if it is durably stored and always available.
- You don't consider the consistency of your cached data.
- You don't monitor the efficiency of your caching implementation.

Benefits of establishing this best practice: Storing data in a cache can improve read latency, read throughput, user experience, and overall efficiency, as well as reduce costs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

A cache is a software or hardware component aimed at storing data so that future requests for the same data can be served faster or more efficiently. The data stored in a cache can be reconstructed if lost by repeating an earlier calculation or fetching it from another data store.

Data caching can be one of the most effective strategies to improve your overall application performance and reduce burden on your underlying primary data sources. Data can be cached at multiple levels in the application, such as within the application making remote calls, known as *client-side caching*, or by using a fast secondary service for storing the data, known as *remote caching*.

Client-side caching

With client-side caching, each client (an application or service that queries the backend datastore) can store the results of their unique queries locally for a specified amount of time. This can reduce the number of requests across the network to a datastore by checking the local client cache first. If the results are not present, the application can then query the datastore and store those results locally. This pattern allows each client to store data in the closest location possible (the client

itself), resulting in the lowest possible latency. Clients can also continue to serve some queries when the backend datastore is unavailable, increasing the availability of the overall system.

One disadvantage of this approach is that when multiple clients are involved, they may store the same cached data locally. This results in both duplicate storage usage and data inconsistency between those clients. One client might cache the results of a query, and one minute later another client can run the same query and get a different result.

Remote caching

To solve the issue of duplicate data between clients, a fast external service, or *remote cache*, can be used to store the queried data. Instead of checking a local data store, each client will check the remote cache before querying the backend datastore. This strategy allows for more consistent responses between clients, better efficiency in stored data, and a higher volume of cached data because the storage space scales independently of clients.

The disadvantage of a remote cache is that the overall system may see a higher latency, as an additional network hop is required to check the remote cache. Client-side caching can be used alongside remote caching for multi-level caching to improve the latency.

Implementation steps

- Identify databases, APIs and network services that could benefit from caching. Services that have heavy read workloads, have a high read-to-write ratio, or are expensive to scale are candidates for caching.
  - Database Caching
  - Enabling API caching to enhance responsiveness
- Identify the appropriate type of caching strategy that best fits your access pattern.
  - Caching strategies
  - AWS Caching Solutions
- Follow Caching Best Practices for your data store.
- Configure a cache invalidation strategy, such as a time-to-live (TTL), for all data that balances freshness of data and reducing pressure on backend datastore.
- Enable features such as automatic connection retries, exponential backoff, client-side timeouts, and connection pooling in the client, if available, as they can improve performance and reliability.
  - Best practices: Redis clients and Amazon ElastiCache (Redis OSS)

- Monitor cache hit rate with a goal of 80% or higher. Lower values may indicate insufficient cache size or an access pattern that does not benefit from caching.
  - Which metrics should I monitor?
  - Best practices for monitoring Redis workloads on Amazon ElastiCache
  - Monitoring best practices with Amazon ElastiCache (Redis OSS) using Amazon CloudWatch
- Implement data replication to offload reads to multiple instances and improve data read performance and availability.

Resources

Related documents:

- Using the Amazon ElastiCache Well-Architected Lens
- Monitoring best practices with Amazon ElastiCache (Redis OSS) using Amazon CloudWatch
- Which Metrics Should I Monitor?
- Performance at Scale with Amazon ElastiCache whitepaper
- Caching challenges and strategies

Related videos:

- Amazon ElastiCache Learning Path
- Design for success with Amazon ElastiCache best practices
- AWS re:Invent 2020 - Design for success with Amazon ElastiCache best practices
- AWS re:Invent 2023 - [LAUNCH] Introducing Amazon ElastiCache Serverless
- AWS re:Invent 2022 - 5 great ways to reimagine your data layer with Redis
- AWS re:Invent 2021 - Deep dive on Amazon ElastiCache (Redis OSS)

Related examples:

- Boosting MySQL database performance with Amazon ElastiCache (Redis OSS)

Networking and content delivery

Questions

- PERF 4. How do you select and configure networking resources in your workload?

PERF 4. How do you select and configure networking resources in your workload?

The optimal networking solution for a workload varies based on latency, throughput requirements, jitter, and bandwidth. Physical constraints, such as user or on-premises resources, determine location options. These constraints can be offset with edge locations or resource placement.

Best practices

- PERF04-BP01 Understand how networking impacts performance
- PERF04-BP02 Evaluate available networking features
- PERF04-BP03 Choose appropriate dedicated connectivity or VPN for your workload
- PERF04-BP04 Use load balancing to distribute traffic across multiple resources
- PERF04-BP05 Choose network protocols to improve performance
- PERF04-BP06 Choose your workload's location based on network requirements
- PERF04-BP07 Optimize network configuration based on metrics

PERF04-BP01 Understand how networking impacts performance

Analyze and understand how network-related decisions impact your workload to provide efficient performance and improved user experience.

Common anti-patterns:

- All traffic flows through your existing data centers.
- You route all traffic through central firewalls instead of using cloud-native network security tools.
- You provision AWS Direct Connect connections without understanding actual usage requirements.
- You don’t consider workload characteristics and encryption overhead when defining your networking solutions.
- You use on-premises concepts and strategies for networking solutions in the cloud.

Benefits of establishing this best practice: Understanding how networking impacts workload performance helps you identify potential bottlenecks, improve user experience, increase reliability, and lower operational maintenance as the workload changes.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The network is responsible for the connectivity between application components, cloud services, edge networks, and on-premises data, and therefore it can heavily impact workload performance. In addition to workload performance, user experience can be also impacted by network latency, bandwidth, protocols, location, network congestion, jitter, throughput, and routing rules.

Have a documented list of networking requirements from the workload including latency, packet size, routing rules, protocols, and supporting traffic patterns. Review the available networking solutions and identify which service meets your workload networking characteristics. Cloud-based networks can be quickly rebuilt, so evolving your network architecture over time is necessary to improve performance efficiency.

Implementation steps:

- Define and document networking performance requirements, including metrics such as network latency, bandwidth, protocols, locations, traffic patterns (spikes and frequency), throughput, encryption, inspection, and routing rules.
- Learn about key AWS networking services like VPCs, AWS Direct Connect, Elastic Load Balancing (ELB), and Amazon Route 53.
- Capture the following key networking characteristics:

Characteristics	Tools and metrics
Foundational networking characteristics	• VPC Flow Logs • AWS Transit Gateway Flow Logs • AWS Transit Gateway metrics • AWS PrivateLink metrics
Application networking characteristics	• Elastic Fabric Adapter • AWS App Mesh metrics • Amazon API Gateway metrics

Characteristics	Tools and metrics
Edge networking characteristics	• Amazon CloudFront metrics • Amazon Route 53 metrics • AWS Global Accelerator metrics
Hybrid networking characteristics	• Direct Connect metrics • AWS Site-to-Site VPN metrics • AWS Client VPN metrics • AWS Cloud WAN metrics
Security networking characteristics	• AWS Shield, AWS WAF, and AWS Network Firewall metrics
Tracing characteristics	• AWS X-Ray • VPC Reachability Analyzer • Network Access Analyzer • Amazon Inspector • Amazon CloudWatch RUM

- Benchmark and test network performance:
  - Benchmark network throughput, as some factors can affect Amazon EC2 network performance when instances are in the same VPC. Measure the network bandwidth between Amazon EC2 Linux instances in the same VPC.
  - Perform load tests to experiment with networking solutions and options.

Resources

Related documents:

- Application Load Balancer
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances

- Network Load Balancer
- Networking Products with AWS
- Transit Gateway
- Transitioning to latency-based routing in Amazon Route 53
- VPC Endpoints

Related videos:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - What can networking do for your application?
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2023 - A developer's guide to cloud networking
- AWS re:Invent 2019 - Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2019 - Optimizing Network Performance for Amazon EC2 Instances
- AWS Summit Online - Improve Global Network Performance for Applications
- AWS re:Invent 2020 - Networking best practices and tips with the Well-Architected Framework
- AWS re:Invent 2020 - AWS networking best practices in large-scale migrations

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops
- Hands-on Network Firewall Workshop
- Observing and Diagnosing your Network on AWS
- Finding and addressing Network Misconfigurations on AWS

PERF04-BP02 Evaluate available networking features

Evaluate networking features in the cloud that may increase performance. Measure the impact of these features through testing, metrics, and analysis. For example, take advantage of network-level features that are available to reduce latency, network distance, or jitter.

Common anti-patterns:

- You stay within one Region because that is where your headquarters is physically located.
- You use firewalls instead of security groups for filtering traffic.
- You break TLS for traffic inspection rather than relying on security groups, endpoint policies, and other cloud-native functionality.
- You only use subnet-based segmentation instead of security groups.

Benefits of establishing this best practice: Evaluating all service features and options can increase your workload performance, reduce the cost of infrastructure, decrease the effort required to maintain your workload, and increase your overall security posture. You can use the global AWS backbone to provide the optimal networking experience for your customers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS offers services like AWS Global Accelerator and Amazon CloudFront that can help improve network performance, while most AWS services have product features (such as the Amazon S3 Transfer Acceleration feature) to optimize network traffic.

Review which network-related configuration options are available to you and how they could impact your workload. Performance optimization depends on understanding how these options interact with your architecture and the impact that they will have on both measured performance and user experience.

Implementation steps

- Create a list of workload components.
  - Consider using AWS Cloud WAN to build, manage and monitor your organization's network when building a unified global network.
  - Monitor your global and core networks with Amazon CloudWatch Logs metrics. Leverage Amazon CloudWatch RUM, which provides insights to help to identify, understand, and enhance users' digital experience.
  - View aggregate network latency between AWS Regions and Availability Zones, as well as within each Availability Zone, using AWS Network Manager to gain insight into how your application performance relates to the performance of the underlying AWS network.
  - Use an existing configuration management database (CMDB) tool or a service such as AWS Config to create an inventory of your workload and how it's configured.

- If this is an existing workload, identify and document the benchmark for your performance metrics, focusing on the bottlenecks and areas to improve. Performance-related networking metrics will differ per workload based on business requirements and workload characteristics. As a start, these metrics might be important to review for your workload: bandwidth, latency, packet loss, jitter, and retransmits.
- If this is a new workload, perform load tests to identify performance bottlenecks.
- For the performance bottlenecks you identify, review the configuration options for your solutions to identify performance improvement opportunities. Check out the following key networking options and features:

Improvement opportunity	Solution
Network path or routes	Use Network Access Analyzer to identify paths or routes.
Network protocols	See PERF04-BP05 Choose network protocols to improve performance
Network topology	Evaluate your operational and performan ce tradeoffs between VPC Peering and AWS Transit Gateway when connecting multiple accounts. AWS Transit Gateway simplifie s how you interconnect all of your VPCs, which can span across thousands of AWS accounts and into on-premises networks. Share your AWS Transit Gateway between multiple accounts using AWS Resource Access Manager. See PERF04-BP03 Choose appropriate dedicated connectivity or VPN for your workload
Network services	AWS Global Accelerator is a networking service that improves the performance of your users' traffic by up to 60% using the AWS global network infrastructure.

Improvement opportunity	Solution
	Amazon CloudFront can improve the performance of your workload content delivery and latency globally. Use Lambda@edge to run functions that customize the content that CloudFront delivers closer to the users, reduce latency, and improve performance. Amazon Route 53 offers latency-based routing, geolocation routing, geoproximity routing, and IP-based routing options to help you improve your workload's performan ce for a global audience. Identify which routing option would optimize your workload performance by reviewing your workload traffic and user location when your workload is distributed globally.
Storage resource features	Amazon S3 Transfer Acceleration is a feature that lets external users benefit from the networking optimizations of CloudFront to upload data to Amazon S3. This improves the ability to transfer large amounts of data from remote locations that don't have dedicated connectivity to the AWS Cloud. Amazon S3 Multi-Region Access Points replicates content to multiple Regions and simplifies the workload by providing one access point. When a Multi-Region Access Point is used, you can request or write data to Amazon S3 with the service identifying the lowest latency bucket.

Improvement opportunity	Solution
Compute resource features	Elastic Network Interfaces (ENI) used by Amazon EC2 instances, containers, and Lambda functions are limited on a per-flow basis. Review your placement groups to optimize your EC2 networking throughput. To avoid a bottleneck on a per flow-basis, design your application to use multiple flows. To monitor and get visibility into your compute related networking metrics, use CloudWatch Metrics and ethtool. The `ethtool` command is included in the ENA driver and exposes additional network-related metrics that can be published as a custom metric to CloudWatch. Amazon Elastic Network Adapters (ENA) provide further optimization by delivering better throughput for your instances within a cluster placement group. Elastic Fabric Adapter (EFA) is a network interface for Amazon EC2 instances that allows you to run workloads requiring high levels of internode communications at scale on AWS. Amazon EBS-optimized instances use an optimized configuration stack and provide additional, dedicated capacity to increase the Amazon EBS I/O.

Resources

Related documents:

- Application Load Balancer

- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- Network Load Balancer
- Networking Products with AWS
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints
- VPC Flow Logs

Related videos:

- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2018 – Optimizing Network Performance for Amazon EC2 Instances
- AWS Global Accelerator

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops
- Observing and diagnosing your network
- Finding and addressing network misconfigurations on AWS

PERF04-BP03 Choose appropriate dedicated connectivity or VPN for your workload

When hybrid connectivity is required to connect on-premises and cloud resources, provision adequate bandwidth to meet your performance requirements. Estimate the bandwidth and latency requirements for your hybrid workload. These numbers will drive your sizing requirements.

Common anti-patterns:

- You only evaluate VPN solutions for your network encryption requirements.
- You do not evaluate backup or redundant connectivity options.
- You do not identify all workload requirements (encryption, protocol, bandwidth, and traffic needs).

Benefits of establishing this best practice: Selecting and configuring appropriate connectivity solutions will increase the reliability of your workload and maximize performance. By identifying workload requirements, planning ahead, and evaluating hybrid solutions, you can minimize expensive physical network changes and operational overhead while increasing your time-to-value.

Level of risk exposed if this best practice is not established: High

Implementation guidance

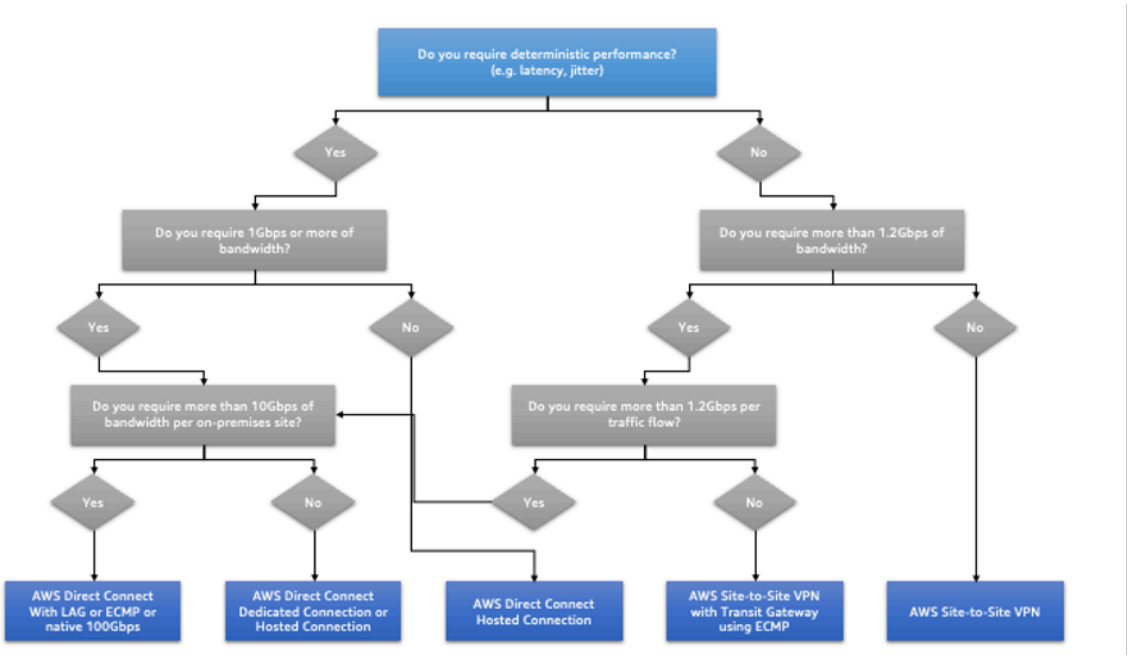
Develop a hybrid networking architecture based on your bandwidth requirements. Direct Connect allows you to connect your on-premises network privately with AWS. It is suitable when you need high-bandwidth and low-latency while achieving consistent performance. A VPN connection establishes secure connection over the internet. It is used when only a temporary connection is required, when cost is a factor, or as a contingency while waiting for resilient physical network connectivity to be established when using Direct Connect.

If your bandwidth requirements are high, you might consider multiple Direct Connect or VPN services. Traffic can be load balanced across services, although we don't recommend load balancing between Direct Connect and VPN because of the latency and bandwidth differences.

Implementation steps

- Estimate the bandwidth and latency requirements of your existing applications.
  - For existing workloads that are moving to AWS, leverage the data from your internal network monitoring systems.
  - For new or existing workloads for which you don’t have monitoring data, consult with the product owners to determine adequate performance metrics and provide a good user experience.
- Select dedicated connection or VPN as your connectivity option. Based on all workload requirements (encryption, bandwidth, and traffic needs), you can either choose AWS Direct Connect or Site-to-Site VPN (or both). The following diagram can help you choose the appropriate connection type.

  - AWS Direct Connect provides dedicated connectivity to the AWS environment, from 50 Mbps up to 100 Gbps, using either dedicated connections or hosted connections. This gives you managed and controlled latency and provisioned bandwidth so your workload can connect efficiently to other environments. Using AWS Direct Connect partners, you can have end-to-end connectivity from multiple environments, providing an extended network with consistent performance. AWS offers scaling direct connect connection bandwidth using either native 100 Gbps, link aggregation group (LAG), or BGP equal-cost multipath (ECMP).
  - The AWS Site-to-Site VPN provides a managed VPN service supporting internet protocol security (IPsec). When a VPN connection is created, each VPN connection includes two tunnels for high availability.
- Follow AWS documentation to choose an appropriate connectivity option:
  - If you decide to use Direct Connect, select the appropriate bandwidth for your connectivity.
  - If you are using an AWS Site-to-Site VPN across multiple locations to connect to an AWS Region, use an accelerated Site-to-Site VPN connection for the opportunity to improve network performance.
  - If your network design consists of IPSec VPN connection over AWS Direct Connect, consider using Private IP VPN to improve security and achieve segmentation. AWS Site-to-Site Private IP VPN is deployed on top of transit virtual interface (VIF).
  - AWS Direct Connect SiteLink allows creating low-latency and redundant connections between your data centers worldwide by sending data over the fastest path between AWS Direct Connect locations, bypassing AWS Regions.
- Validate your connectivity setup before deploying to production. Perform security and performance testing to assure it meets your bandwidth, reliability, latency, and compliance requirements.
- Regularly monitor your connectivity performance and usage and optimize if required.



Deterministic performance flowchart

Resources

Related documents:

- Networking Products with AWS
- AWS Transit Gateway
- VPC Endpoints
- Building a Scalable and Secure Multi-VPC AWS Network Infrastructure
- Client VPN

Related videos:

- AWS re:Invent 2023 – Building hybrid network connectivity with AWS
- AWS re:Invent 2023 – Secure remote connectivity to AWS
- AWS re:Invent 2022 – Optimizing performance with Amazon CloudFront
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2020 – AWS Transit Gateway Connect

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP04 Use load balancing to distribute traffic across multiple resources

Distribute traffic across multiple resources or services to allow your workload to take advantage of the elasticity that the cloud provides. You can also use load balancing for offloading encryption termination to improve performance, reliability and manage and route traffic effectively.

Common anti-patterns:

- You don't consider your workload requirements when choosing the load balancer type.
- You don't leverage the load balancer features for performance optimization.
- The workload is exposed directly to the internet without a load balancer.
- You route all internet traffic through existing load balancers.
- You use generic TCP load balancing and making each compute node handle SSL encryption.

Benefits of establishing this best practice: A load balancer handles the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones and enables high availability, automatic scaling, and better utilization for your workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Load balancers act as the entry point for your workload, from which point they distribute the traffic to your backend targets, such as compute instances or containers, to improve utilization.

Choosing the right load balancer type is the first step to optimize your architecture. Start by listing your workload characteristics, such as protocol (like TCP, HTTP, TLS, or WebSockets), target type (like instances, containers, or serverless), application requirements (like long running connections, user authentication, or stickiness), and placement (like Region, Local Zone, Outpost, or zonal isolation).

AWS provides multiple models for your applications to use load balancing. Application Load Balancer is best suited for load balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers.

Network Load Balancer is best suited for load balancing of TCP traffic where extreme performance is required. It is capable of handling millions of requests per second while maintaining ultra-low latencies, and it is optimized to handle sudden and volatile traffic patterns.

Elastic Load Balancing provides integrated certificate management and SSL/TLS decryption, allowing you the flexibility to centrally manage the SSL settings of the load balancer and offload CPU intensive work from your workload.

After choosing the right load balancer, you can start leveraging its features to reduce the amount of effort your backend has to do to serve the traffic.

For example, using both Application Load Balancer (ALB) and Network Load Balancer (NLB), you can perform SSL/TLS encryption offloading, which is an opportunity to avoid the CPU-intensive TLS handshake from being completed by your targets and also to improve certificate management.

When you configure SSL/TLS offloading in your load balancer, it becomes responsible for the encryption of the traffic from and to clients while delivering the traffic unencrypted to your backends, freeing up your backend resources and improving the response time for the clients.

Application Load Balancer can also serve HTTP/2 traffic without needing to support it on your targets. This simple decision can improve your application response time, as HTTP/2 uses TCP connections more efficiently.

Your workload latency requirements should be considered when defining the architecture. As an example, if you have a latency-sensitive application, you may decide to use Network Load Balancer, which offers extremely low latencies. Alternatively, you may decide to bring your workload closer to your customers by leveraging Application Load Balancer in AWS Local Zones or even AWS Outposts.

Another consideration for latency-sensitive workloads is cross-zone load balancing. With cross-zone load balancing, each load balancer node distributes traffic across the registered targets in all allowed Availability Zones.

Use Auto Scaling integrated with your load balancer. One of the key aspects of a performance efficient system has to do with right-sizing your backend resources. To do this, you can leverage load balancer integrations for backend target resources. Using the load balancer integration with Auto Scaling groups, targets will be added or removed from the load balancer as required in response to incoming traffic. Load balancers can also integrate with Amazon ECS and Amazon EKS for containerized workloads.

- Amazon ECS - Service load balancing

- Application load balancing on Amazon EKS
- Network load balancing on Amazon EKS

Implementation steps

- Define your load balancing requirements including traffic volume, availability and application scalability.
- Choose the right load balancer type for your application.
  - Use Application Load Balancer for HTTP/HTTPS workloads.
  - Use Network Load Balancer for non-HTTP workloads that run on TCP or UDP.
  - Use a combination of both (ALB as a target of NLB) if you want to leverage features of both products. For example, you can do this if you want to use the static IPs of NLB together with HTTP header based routing from ALB, or if you want to expose your HTTP workload to an AWS PrivateLink.
  - For a full comparison of load balancers, see ELB product comparison.
- Use SSL/TLS offloading if possible.
  - Configure HTTPS/TLS listeners with both Application Load Balancer and Network Load Balancer integrated with AWS Certificate Manager.
  - Note that some workloads may require end-to-end encryption for compliance reasons. In this case, it is a requirement to allow encryption at the targets.
  - For security best practices, see SEC09-BP02 Enforce encryption in transit.
- Select the right routing algorithm (only ALB).
  - The routing algorithm can make a difference in how well-used your backend targets are and therefore how they impact performance. For example, ALB provides two options for routing algorithms:
  - **Least outstanding requests:** Use to achieve a better load distribution to your backend targets for cases when the requests for your application vary in complexity or your targets vary in processing capability.
  - **Round robin:** Use when the requests and targets are similar, or if you need to distribute requests equally among targets.
- Consider cross-zone or zonal isolation.
  - Use cross-zone turned off (zonal isolation) for latency improvements and zonal failure domains. It is turned off by default in NLB and in ALB you can turn it off per target group.

  - Use cross-zone turned on for increased availability and flexibility. By default, cross-zone is turned on for ALB and in NLB you can turn it on per target group.
- Turn on HTTP keep-alives for your HTTP workloads (only ALB). With this feature, the load balancer can reuse backend connections until the keep-alive timeout expires, improving your HTTP request and response time and also reducing resource utilization on your backend targets. For detail on how to do this for Apache and Nginx, see What are the optimal settings for using Apache or NGINX as a backend server for ELB?
- Turn on monitoring for your load balancer.
  - Turn on access logs for your Application Load Balancer and Network Load Balancer.
  - The main fields to consider for ALB are `request_processing_time`, `request_processing_time`, and `response_processing_time`.
  - The main fields to consider for NLB are `connection_time` and `tls_handshake_time`.
  - Be ready to query the logs when you need them. You can use Amazon Athena to query both ALB logs and NLB logs.
  - Create alarms for performance related metrics such as `TargetResponseTime` for ALB.

Resources

Related documents:

- ELB product comparison
- AWS Global Infrastructure
- Improving Performance and Reducing Cost Using Availability Zone Affinity
- Step by step for Log Analysis with Amazon Athena
- Querying Application Load Balancer logs
- Monitor your Application Load Balancers
- Monitor your Network Load Balancer
- Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group

Related videos:

- AWS re:Invent 2023: What can networking do for your application?
- AWS re:Inforce 20: How to use Elastic Load Balancing to enhance your security posture at scale

- AWS re:Invent 2018: Elastic Load Balancing: Deep Dive and Best Practices
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads

Related examples:

- Gateway Load Balancer
- CDK and CloudFormation samples for Log Analysis with Amazon Athena

PERF04-BP05 Choose network protocols to improve performance

Make decisions about protocols for communication between systems and networks based on the impact to the workload's performance.

There is a relationship between latency and bandwidth to achieve throughput. If your file transfer is using Transmission Control Protocol (TCP), higher latencies will most likely reduce overall throughput. There are approaches to fix this with TCP tuning and optimized transfer protocols, but one solution is to use User Datagram Protocol (UDP).

Common anti-patterns:

- You use TCP for all workloads regardless of performance requirements.

Benefits of establishing this best practice: Verifying that an appropriate protocol is used for communication between users and workload components helps improve overall user experience for your applications. For instance, connection-less UDP allows for high speed, but it doesn't offer retransmission or high reliability. TCP is a full featured protocol, but it requires greater overhead for processing the packets.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

If you have the ability to choose different protocols for your application and you have expertise in this area, optimize your application and end-user experience by using a different protocol. Note that this approach comes with significant difficulty and should only be attempted if you have optimized your application in other ways first.

A primary consideration for improving your workload's performance is to understand the latency and throughput requirements, and then choose network protocols that optimize performance.

When to consider using TCP

TCP provides reliable data delivery, and can be used for communication between workload components where reliability and guaranteed delivery of data is important. Many web-based applications rely on TCP-based protocols, such as HTTP and HTTPS, to open TCP sockets for communication between application components. Email and file data transfer are common applications that also make use of TCP, as it is a simple and reliable transfer mechanism between application components. Using TLS with TCP can add some overhead to the communication, which can result in increased latency and reduced throughput, but it comes with the advantage of security. The overhead comes mainly from the added overhead of the handshake process, which can take several round-trips to complete. Once the handshake is complete, the overhead of encrypting and decrypting data is relatively small.

When to consider using UDP

UDP is a connection-less-oriented protocol and is therefore suitable for applications that need fast, efficient transmission, such as log, monitoring, and VoIP data. Also, consider using UDP if you have workload components that respond to small queries from large numbers of clients to ensure optimal performance of the workload. Datagram Transport Layer Security (DTLS) is the UDP equivalent of Transport Layer Security (TLS). When using DTLS with UDP, the overhead comes from encrypting and decrypting the data, as the handshake process is simplified. DTLS also adds a small amount of overhead to the UDP packets, as it includes additional fields to indicate the security parameters and to detect tampering.

When to consider using SRD

Scalable reliable datagram (SRD) is a network transport protocol optimized for high-throughput workloads due to its ability to load-balancer traffic across multiple paths and quickly recover from packet drops or link failures. SRD is therefore best used for high performance computing (HPC) workloads that require high throughput and low latency communication between compute nodes. This might include parallel processing tasks such as simulation, modeling, and data analysis that involve a large amount of data transfer between nodes.

Implementation steps

- Use the AWS Global Accelerator and AWS Transfer Family services to improve the throughput of your online file transfer applications. The AWS Global Accelerator service helps you achieve

lower latency between your client devices and your workload on AWS. With AWS Transfer Family, you can use TCP-based protocols such as Secure Shell File Transfer Protocol (SFTP) and File Transfer Protocol over SSL (FTPS) to securely scale and manage your file transfers to AWS storage services.

- Use network latency to determine if TCP is appropriate for communication between workload components. If the network latency between your client application and server is high, then the TCP three-way handshake can take some time, thereby impacting on the responsiveness of your application. Metrics such as time to first byte (TTFB) and round-trip time (RTT) can be used to measure network latency. If your workload serves dynamic content to users, consider using Amazon CloudFront, which establishes a persistent connection to each origin for dynamic content to remove the connection setup time that would otherwise slow down each client request.
- Using TLS with TCP or UDP can result in increased latency and reduced throughput for your workload due to the impact of encryption and decryption. For such workloads, consider SSL/TLS offloading on Elastic Load Balancing to improve workload performance by allowing the load balancer to handle SSL/TLS encryption and decryption process instead of having backend instances do it. This can help reduce the CPU utilization on the backend instances, which can improve performance and increase capacity.
- Use the Network Load Balancer (NLB) to deploy services that rely on the UDP protocol, such as authentication and authorization, logging, DNS, IoT, and streaming media, to improve the performance and reliability of your workload. The NLB distributes incoming UDP traffic across multiple targets, allowing you to scale your workload horizontally, increase capacity, and reduce the overhead of a single target.
- For your High Performance Computing (HPC) workloads, consider using the Elastic Network Adapter (ENA) Express functionality that uses the SRD protocol to improve network performance by providing a higher single flow bandwidth (25 Gbps) and lower tail latency (99.9 percentile) for network traffic between EC2 instances.
- Use the Application Load Balancer (ALB) to route and load balance your gRPC (Remote Procedure Calls) traffic between workload components or between gRPC clients and services. gRPC uses the TCP-based HTTP/2 protocol for transport and it provides performance benefits such as lighter network footprint, compression, efficient binary serialization, support for numerous languages, and bi-directional streaming.

Resources

Related documents:

- How to route UDP traffic into Kubernetes
- Application Load Balancer
- EC2 Enhanced Networking on Linux
- EC2 Enhanced Networking on Windows
- EC2 Placement Groups
- Enabling Enhanced Networking with the Elastic Network Adapter (ENA) on Linux Instances
- Network Load Balancer
- Networking Products with AWS
- Transitioning to Latency-Based Routing in Amazon Route 53
- VPC Endpoints

Related videos:

- AWS re:Invent 2022 – Scaling network performance on next-gen Amazon Elastic Compute Cloud instances
- AWS re:Invent 2022 – Application networking foundations

Related examples:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP06 Choose your workload's location based on network requirements

Evaluate options for resource placement to reduce network latency and improve throughput, providing an optimal user experience by reducing page load and data transfer times.

Common anti-patterns:

- You consolidate all workload resources into one geographic location.
- You chose the closest Region to your location but not to the workload end user.

Benefits of establishing this best practice: User experience is greatly affected by the latency between the user and your application. By using appropriate AWS Regions and the AWS private global network, you can reduce latency and deliver a better experience to remote users.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Resources, such as Amazon EC2 instances, are placed into Availability Zones within AWS Regions, AWS Local Zones, AWS Outposts, or AWS Wavelength zones. Selection of this location influences network latency and throughput from a given user location. Edge services like Amazon CloudFront and AWS Global Accelerator can also be used to improve network performance by either caching content at edge locations or providing users with an optimal path to the workload through the AWS global network.

Amazon EC2 provides placement groups for networking. A placement group is a logical grouping of instances to decrease latency. Using placement groups with supported instance types and an Elastic Network Adapter (ENA) enables workloads to participate in a low-latency, reduced jitter 25 Gbps network. Placement groups are recommended for workloads that benefit from low network latency, high network throughput, or both.

Latency-sensitive services are delivered at edge locations using AWS global network, such as Amazon CloudFront. These edge locations commonly provide services like content delivery network (CDN) and domain name system (DNS). By having these services at the edge, workloads can respond with low latency to requests for content or DNS resolution. These services also provide geographic services, such as geotargeting of content (providing different content based on the end users' location) or latency-based routing to direct end users to the nearest Region (minimum latency).

Use edge services to reduce latency and to enable content caching. Configure cache control correctly for both DNS and HTTP/HTTPS to gain the most benefit from these approaches.

Implementation steps

- Capture information about the IP traffic going to and from network interfaces.
  - Logging IP traffic using VPC Flow Logs
  - How the client IP address is preserved in AWS Global Accelerator
- Analyze network access patterns in your workload to identify how users use your application.
  - Use monitoring tools, such as Amazon CloudWatch and AWS CloudTrail, to gather data on network activities.
  - Analyze the data to identify the network access pattern.
- Select Regions for your workload deployment based on the following key elements:

  - **Where your data is located:** For data-heavy applications (such as big data and machine learning), application code should run as close to the data as possible.
  - **Where your users are located**: For user-facing applications, choose a Region (or Regions) close to your workload's users.
  - **Other constraints**: Consider constraints such as cost and compliance as explained in What to Consider when Selecting a Region for your Workloads.
- Use AWS Local Zones to run workloads like video rendering. Local Zones allow you to benefit from having compute and storage resources closer to end users.
- Use AWS Outposts for workloads that need to remain on-premises and where you want that workload to run seamlessly with the rest of your other workloads in AWS.
- Applications like high-resolution live video streaming, high-fidelity audio, and augmented reality or virtual reality (AR/VR) require ultra-low-latency for 5G devices. For such applications, consider AWS Wavelength. AWS Wavelength embeds AWS compute and storage services within 5G networks, providing mobile edge computing infrastructure for developing, deploying, and scaling ultra-low-latency applications.
- Use local caching or AWS Caching Solutions for frequently used assets to improve performance, reduce data movement, and lower environmental impact.

Service	When to use
Amazon CloudFront	Use to cache static content such as images, scripts, and videos, as well as dynamic content such as API responses or web applications.
Amazon ElastiCache	Use to cache content for web applications.
DynamoDB Accelerator	Use to add in-memory acceleration to your DynamoDB tables.

- Use services that can help you run code closer to users of your workload like the following:

Service	When to use
Lambda@edge	Use for compute-heavy operations that are initiated when objects are not in the cache.

Service	When to use
Amazon CloudFront Functions	Use for simple use cases like HTTP(s) requests or response manipulations that can be initiated by short-lived functions.
AWS IoT Greengrass	Use to run local compute, messaging, and data caching for connected devices.

- Some applications require fixed entry points or higher performance by reducing first byte latency and jitter, and increasing throughput. These applications can benefit from networking services that provide static anycast IP addresses and TCP termination at edge locations. AWS Global Accelerator can improve performance for your applications by up to 60% and provide quick failover for multi-region architectures. AWS Global Accelerator provides you with static anycast IP addresses that serve as a fixed entry point for your applications hosted in one or more AWS Regions. These IP addresses permit traffic to ingress onto the AWS global network as close to your users as possible. AWS Global Accelerator reduces the initial connection setup time by establishing a TCP connection between the client and the AWS edge location closest to the client. Review the use of AWS Global Accelerator to improve the performance of your TCP/UDP workloads and provide quick failover for multi-Region architectures.

Resources

Related best practices:

- COST07-BP02 Implement Regions based on cost
- COST08-BP03 Implement services to reduce data transfer costs
- REL10-BP01 Deploy the workload to multiple locations
- REL10-BP02 Select the appropriate locations for your multi-location deployment
- SUS01-BP01 Choose Region based on both business requirements and sustainability goals
- SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements
- SUS04-BP07 Minimize data movement across networks

Related documents:

- AWS Global Infrastructure
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- Placement groups
- AWS Local Zones
- AWS Outposts
- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- AWS Site-to-Site VPN
- Amazon Route 53

Related videos:

- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2023 - A migration strategy for edge and on-premises workloads
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

Related examples:

- AWS Global Accelerator Custom Routing Workshop
- Handling Rewrites and Redirects using Edge Functions

PERF04-BP07 Optimize network configuration based on metrics

Use collected and analyzed data to make informed decisions about optimizing your network configuration.

Common anti-patterns:

- You assume that all performance-related issues are application-related.
- You only test your network performance from a location close to where you have deployed the workload.
- You use default configurations for all network services.
- You overprovision the network resource to provide sufficient capacity.

Benefits of establishing this best practice: Collecting necessary metrics of your AWS network and implementing network monitoring tools allows you to understand network performance and optimize network configurations.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Monitoring traffic to and from VPCs, subnets, or network interfaces is crucial to understand how to utilize AWS network resources and optimize network configurations. By using the following AWS networking tools, you can further inspect information about the traffic usage, network access and logs.

Implementation steps

- Identify the key performance metrics such as latency or packet loss to collect. AWS provides several tools that can help you to collect these metrics. By using the following tools, you can further inspect information about the traffic usage, network access, and logs:

AWS tool	Where to use
Amazon VPC IP Address Manager.	Use IPAM to plan, track, and monitor IP addresses for your AWS and on-premises workloads. This is a best practice to optimize IP address usage and allocation.

AWS tool	Where to use
VPC Flow logs	Use VPC Flow Logs to capture detailed information about traffic to and from network interfaces in your VPCs. With VPC Flow Logs, you can diagnose overly restricti ve or permissive security group rules and determine the direction of the traffic to and from the network interfaces.
AWS Transit Gateway Flow Logs	Use AWS Transit Gateway Flow Logs to capture information about the IP traffic going to and from your transit gateways.
DNS query logging	Log information about public or private DNS queries Route 53 receives. With DNS logs, you can optimize DNS configurations by understanding the domain or subdomain that was requested or Route 53 EDGE locations that responded to DNS queries.
Reachability Analyzer	Reachability Analyzer helps you analyze and debug network reachability. Reachability Analyzer is a configuration analysis tool that allows you to perform connectivity testing between a source resource and a destinati on resource in your VPCs. This tool helps you verify that your network configuration matches your intended connectivity.

AWS tool	Where to use
Network Access Analyzer	Network Access Analyzer helps you understand network access to your resources. You can use Network Access Analyzer to specify your network access requirements and identify potential network paths that do not meet your specified requirements. By optimizing your corresponding network configuration, you can understand and verify the state of your network and demonstrate if your network on AWS meets your compliance requirements.
Amazon CloudWatch	Use Amazon CloudWatch and turn on the appropriate metrics for network options. Make sure to choose the right network metric for your workload. For example, you can turn on metrics for VPC Network Address Usage, VPC NAT Gateway, AWS Transit Gateway, VPN tunnel, AWS Network Firewall, Elastic Load Balancing, and AWS Direct Connect. Continually monitoring metrics is a good practice to observe and understand your network status and usage, which helps you optimize network configuration based on your observations.

AWS tool	Where to use
AWS Network Manager	Using AWS Network Manager, you can monitor the real-time and historical performance of the AWS Global Network for operational and planning purposes. Network Manager provides aggregate network latency between AWS Regions and Availability Zones and within each Availability Zone, allowing you to better understand how your applicati on performance relates to the performance of the underlying AWS network.
Amazon CloudWatch RUM	Use Amazon CloudWatch RUM to collect the metrics that give you the insights that help you identify, understand, and improve user experience.

- Identify top talkers and application traffic patterns using VPC and AWS Transit Gateway Flow Logs.
- Assess and optimize your current network architecture including VPCs, subnets, and routing. As an example, you can evaluate how different VPC peering or AWS Transit Gateway can help you improve the networking in your architecture.
- Assess the routing paths in your network to verify that the shortest path between destinations is always used. Network Access Analyzer can help you do this.

Resources

Related documents:

- Public DNS query logging
- What is IPAM?
- What is Reachability Analyzer?
- What is Network Access Analyzer?
- CloudWatch metrics for your VPCs

- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format
- Monitoring your global and core networks with Amazon CloudWatch metrics
- Continuously monitor network traffic and resources

Related videos:

- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2020 – Networking best practices and tips with the AWS Well-Architected Framework
- AWS re:Invent 2020 – Monitoring and troubleshooting network traffic

Related examples:

- AWS Networking Workshops
- AWS Network Monitoring
- Observing and diagnosing your network on AWS
- Finding and addressing network misconfigurations on AWS

Process and culture

Questions

- PERF 5. How do your organizational practices and culture contribute to performance efficiency in your workload?

PERF 5. How do your organizational practices and culture contribute to performance efficiency in your workload?

When architecting workloads, there are principles and practices that you can adopt to help you better run efficient high-performing cloud workloads. To adopt a culture that fosters performance efficiency of cloud workloads, consider these key principles and practices:

Best practices

- PERF05-BP01 Establish key performance indicators (KPIs) to measure workload health and performance
- PERF05-BP02 Use monitoring solutions to understand the areas where performance is most critical
- PERF05-BP03 Define a process to improve workload performance
- PERF05-BP04 Load test your workload
- PERF05-BP05 Use automation to proactively remediate performance-related issues
- PERF05-BP06 Keep your workload and services up-to-date
- PERF05-BP07 Review metrics at regular intervals

PERF05-BP01 Establish key performance indicators (KPIs) to measure workload health and performance

Identify the KPIs that quantitatively and qualitatively measure workload performance. KPIs help you measure the health and performance of a workload related to a business goal.

Common anti-patterns:

- You only monitor system-level metrics to gain insight into your workload and don't understand business impacts to those metrics.
- You assume that your KPIs are already being published and shared as standard metric data.
- You do not define a quantitative, measurable KPI.
- You do not align KPIs with business goals or strategies.

Benefits of establishing this best practice: Identifying specific KPIs that represent workload health and performance helps align teams on their priorities and define successful business outcomes. Sharing those metrics with all departments provides visibility and alignment on thresholds, expectations, and business impact.

Level of risk exposed if this best practice is not established: High

Implementation guidance

KPIs allow business and engineering teams to align on the measurement of goals and strategies and how these factors combine to produce business outcomes. For example, a website workload

might use page load time as an indication of overall performance. This metric would be one of multiple data points that measures user experience. In addition to identifying the page load time thresholds, you should document the expected outcome or business risk if ideal performance is not met. A long page load time affects your end users directly, decreases their user experience rating, and can lead to a loss of customers. When you define your KPI thresholds, combine both industry benchmarks and your end user expectations. For example, if the current industry benchmark is a webpage loading within a two-second time period, but your end users expect a webpage to load within a one-second time period, then you should take both of these data points into consideration when establishing the KPI.

Your team must evaluate your workload KPIs using real-time granular data and historical data for reference and create dashboards that perform metric math on your KPI data to derive operational and utilization insights. KPIs should be documented and include thresholds that support business goals and strategies, and should be mapped to metrics being monitored. KPIs should be revisited when business goals, strategies, or end user requirements change.

Implementation steps

- **Identify stakeholders:** Identify and document key business stakeholders, including development and operation teams.
- **Define objectives:** Work with these stakeholders to define and document objectives of your workload. Consider the critical performance aspects of your workloads, such as throughput, response time, and cost, as well as business goals, such as user satisfaction.
- **Review industry best practices:** Review industry best practices to identify relevant KPIs aligned with your workload objectives.
- **Identify metrics:** Identify metrics that are aligned with your workload objectives and can help you measure performance and business goals. Establish KPIs based on these metrics. Example metrics are measurements like average response time or number of concurrent users.
- **Define and document KPIs:** Use industry best practices and your workload objectives to set targets for your workload KPI. Use this information to set KPI thresholds for severity or alarm level. Identify and document the risk and impact of a KPI is not met.
- **Implement monitoring:** Use monitoring tools such as Amazon CloudWatch or AWS Config to collect metrics and measure KPIs.
- **Visually communicate KPIs:** Use dashboard tools like Amazon Quick to visualize and communicate KPIs with stakeholders.

- **Analyze and optimize:** Regularly review and analyze KPIs to identify areas of your workload that need to be improved. Work with stakeholders to implement these improvements.
- **Revisit and refine:** Regularly review metrics and KPIs to assess their effectiveness, especially when business goals or workload performance change.

Resources

Related documents:

- CloudWatch documentation
- Monitoring, Logging, and Performance AWS Partners
- AWS observability tools
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations
- How to track your cost optimization KPIs with the KPI Dashboard
- X-Ray Documentation
- Using Amazon CloudWatch dashboards
- Quick KPIs

Related videos:

- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health
- AWS re:Invent 2023 - Performance & efficiency at Pinterest: Optimizing the latest instances
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Creating an Effective Metrics Strategy for Your Business | AWS Events

Related examples:

- Creating a dashboard with Quick

PERF05-BP02 Use monitoring solutions to understand the areas where performance is most critical

Understand and identify areas where increasing the performance of your workload will have a positive impact on efficiency or customer experience. For example, a website that has a large amount of customer interaction can benefit from using edge services to move content delivery closer to customers.

Common anti-patterns:

- You assume that standard compute metrics such as CPU utilization or memory pressure are enough to catch performance issues.
- You only use the default metrics recorded by your selected monitoring software.
- You only review metrics when there is an issue.

Benefits of establishing this best practice: Understanding critical areas of performance helps workload owners monitor KPIs and prioritize high-impact improvements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Set up end-to-end tracing to identify traffic patterns, latency, and critical performance areas. Monitor your data access patterns for slow queries or poorly fragmented and partitioned data. Identify the constrained areas of the workload using load testing or monitoring.

Increase performance efficiency by understanding your architecture, traffic patterns, and data access patterns, and identify your latency and processing times. Identify the potential bottlenecks that might affect the customer experience as the workload grows. After investigating these areas, look at which solution you could deploy to remove those performance concerns.

Implementation steps

- Set up end-to-end monitoring to capture all workload components and metrics. Here are examples of monitoring solutions on AWS.

Service	Where to use
Amazon CloudWatch Real-User Monitoring (RUM)	To capture application performance metrics from real user client-side and frontend sessions.
AWS X-Ray	To trace traffic through the application layers and identify latency between components and dependencies. Use X-Ray service maps to see relationships and latency between workload components.
Amazon Relational Database Service Performance Insights	To view database performance metrics and identify performance improvements.
Amazon RDS Enhanced Monitoring	To view database OS performance metrics.
Amazon DevOps Guru	To detect abnormal operating patterns so you can identify operational issues before they impact your customers.

- Perform tests to generate metrics, identify traffic patterns, bottlenecks, and critical performance areas. Here are some examples of how to perform testing:
  - Set up CloudWatch Synthetic Canaries to mimic browser-based user activities programmatically using Linux cron jobs or rate expressions to generate consistent metrics over time.
  - Use the AWS Distributed Load Testing solution to generate peak traffic or test the workload at the expected growth rate.
- Evaluate the metrics and telemetry to identify your critical performance areas. Review these areas with your team to discuss monitoring and solutions to avoid bottlenecks.
- Experiment with performance improvements and measure those changes with data. As an example, you can use CloudWatch Evidently to test new improvements and performance impacts to your workload.

Resources

Related documents:

- What's new in AWS Observability at re:Invent 2023
- Amazon Builders' Library
- X-Ray Documentation
- Amazon CloudWatch RUM
- Amazon DevOps Guru

Related videos:

- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 years of Amazon operational excellence
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Visual Monitoring of Applications with Amazon CloudWatch Synthetics

Related examples:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Python
- Distributed Load Testing on AWS

PERF05-BP03 Define a process to improve workload performance

Define a process to evaluate new services, design patterns, resource types, and configurations as they become available. For example, run existing performance tests on new instance offerings to determine their potential to improve your workload.

Common anti-patterns:

- You assume your current architecture is static and won't be updated over time.
- You introduce architecture changes over time with no metric justification.

Benefits of establishing this best practice: By defining your process for making architectural changes, you can use gathered data to influence your workload design over time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Your workload's performance has a few key constraints. Document these so that you know what kinds of innovation might improve the performance of your workload. Use this information when learning about new services or technology as it becomes available to identify ways to alleviate constraints or bottlenecks.

Identify the key performance constraints for your workload. Document your workload's performance constraints so that you know what kinds of innovation might improve the performance of your workload.

Implementation steps

- **Identify KPIs:** Identify your workload performance KPIs as outlined in PERF05-BP01 Establish key performance indicators (KPIs) to measure workload health and performance to baseline your workload.
- **Implement monitoring:** Use AWS observability tools to collect performance metrics and measure KPIs.
- **Conduct analysis:** Conduct in-depth analysis to identify the areas (like configuration and application code) in your workload that is under-performing as outlined in PERF05-BP02 Use monitoring solutions to understand the areas where performance is most critical. Use your analysis and performance tools to identify the performance improvement strategies.
- **Validate improvements:** Use sandbox or pre-production environments to validate the effectiveness of improvement strategies.
- **Implement changes:** Implement the changes in production and continually monitor the workload's performance. Document the improvements, and communicate the changes to stakeholders.

- **Revisit and refine:** Regularly review your performance improvement process to identify areas for enhancement.

Resources

Related documents:

- AWS Blog
- What's New with AWS
- AWS Skill Builder

Related videos:

- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance

Related examples:

- AWS Github

PERF05-BP04 Load test your workload

Load test your workload to verify it can handle production load and identify any performance bottleneck.

Common anti-patterns:

- You load test individual parts of your workload but not your entire workload.
- You load test on infrastructure that is not the same as your production environment.
- You only conduct load testing to your expected load and not beyond, to help foresee where you may have future problems.
- You perform load testing without consulting the Amazon EC2 Testing Policy and submitting a Simulated Event Submissions Form. This results in your test failing to run, as it looks like a denial-of-service event.

Benefits of establishing this best practice: Measuring your performance under a load test will show you where you will be impacted as load increases. This can provide you with the capability of anticipating needed changes before they impact your workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Load testing in the cloud is a process to measure the performance of cloud workload under realistic conditions with expected user load. This process involves provisioning a production-like cloud environment, using load testing tools to generate load, and analyzing metrics to assess the ability of your workload handling a realistic load. Load tests must be run using synthetic or sanitized versions of production data (remove sensitive or identifying information). Automatically carry out load tests as part of your delivery pipeline, and compare the results against pre-defined KPIs and thresholds. This process helps you continue to achieve required performance.

Implementation steps

- **Define your testing objectives:** Identify the performance aspects of your workload that you want to evaluate, such as throughput and response time.
- **Select a testing tool:** Choose and configure the load testing tool that suits your workload.
- **Set up your environment:** Set up the test environment based on your production environment. You can use AWS services to run production-scale environments to test your architecture.
- **Implement monitoring:** Use monitoring tools such as Amazon CloudWatch to collect metrics across the resources in your architecture. You can also collect and publish custom metrics.
- **Define scenarios:** Define the load testing scenarios and parameters (like test duration and number of users).
- **Conduct load testing:** Perform test scenarios at scale. Take advantage of the AWS Cloud to test your workload to discover where it fails to scale, or if it scales in a non-linear way. For example, use Spot Instances to generate loads at low cost and discover bottlenecks before they are experienced in production.
- **Analyze test results:** Analyze the results to identify performance bottlenecks and areas for improvements.
- **Document and share findings:** Document and report on findings and recommendations. Share this information with stakeholders to help them make informed decision regarding performance optimization strategies.

- **Continually iterate:** Load testing should be performed at regular cadence, especially after a system change of update.

Resources

Related documents:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Distributed Load Testing on AWS

Related videos:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing
- AWS re:Invent 2022 - Scaling on AWS for your first 10 million users
- Solving with AWS Solutions: Distributed Load Testing

- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM
- Demo of Amazon CloudWatch Synthetics

Related examples:

- Distributed Load Testing on AWS

PERF05-BP05 Use automation to proactively remediate performance-related issues

Use key performance indicators (KPIs), combined with monitoring and alerting systems, to proactively address performance-related issues.

Common anti-patterns:

- You only allow operations staff the ability to make operational changes to the workload.
- You let all alarms filter to the operations team with no proactive remediation.

Benefits of establishing this best practice: Proactive remediation of alarm actions allows support staff to concentrate on those items that are not automatically actionable. This helps operations staff handle all alarms without being overwhelmed and instead focus only on critical alarms.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use alarms to trigger automated actions to remediate issues where possible. Escalate the alarm to those able to respond if automated response is not possible. For example, you may have a system that can predict expected key performance indicator (KPI) values and alarm when they breach certain thresholds, or a tool that can automatically halt or roll back deployments if KPIs are outside of expected values.

Implement processes that provide visibility into performance as your workload is running. Build monitoring dashboards and establish baseline norms for performance expectations to determine if the workload is performing optimally.

Implementation steps

- **Identify remediation workflow:** Identify and understand the performance issue that can be remediated automatically. Use AWS monitoring solutions such as Amazon CloudWatch or AWS X-Ray to help you better understand the root cause of the issue.
- **Define the automation process:** Create a step-by-step remediation process that can be used to automatically fix the issue.
- **Configure the initiation event:** Configure the event to automatically initiate the remediation process. For example, you can define a trigger to automatically restart an instance when it reaches a certain threshold of CPU utilization.
- **Automate the remediation:** Use AWS services and technologies to automate the remediation process. For example, AWS Systems Manager Automation provides a secure and scalable way to automate the remediation process. Make sure to use self-healing logic to revert changes if they do not successfully resolve the issue.
- **Test the workflow** Test the automated remediation process in a pre-production environment.
- **Implement the workflow:** Implement the automated remediation in the production environment.
- **Develop a playbook:** Develop and document a playbook that outlines the steps for the remediation plan, including the initiation events, remediation logic, and actions taken. Make sure to train stakeholders to help them effectively respond to automated remediation events.

- **Review and refine:** Regularly assess the effectiveness of the automated remediation workflow. Adjust initiation events and remediation logic if necessary.

Resources

Related documents:

- CloudWatch Documentation
- Monitoring, Logging, and Performance AWS Partner Network Partners
- X-Ray Documentation
- Using Alarms and Alarm Actions in CloudWatch
- Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- Automate your Amazon Redshift performance tuning with automatic table optimization

Related videos:

- AWS re:Invent 2023 - Strategies for automated scaling, remediation, and smart self-healing
- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2021 - Intelligently automating cloud operations
- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - Automating patch management and compliance using AWS
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS
- AWS re:Invent 2021 -{New Launch} Automatically detect and resolve issues with Amazon DevOps Guru
- AWS re:Invent 2023 - Centralize your operations

Related examples:

- CloudWatch Logs Customize Alarms

PERF05-BP06 Keep your workload and services up-to-date

Stay up-to-date on new cloud services and features to adopt efficient features, remove issues, and improve the overall performance efficiency of your workload.

Common anti-patterns:

- You assume your current architecture is static and will not be updated over time.
- You do not have any systems or a regular cadence to evaluate if updated software and packages are compatible with your workload.

Benefits of establishing this best practice: By establishing a process to stay up-to-date on new services and offerings, you can adopt new features and capabilities, resolve issues, and improve workload performance.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Evaluate ways to improve performance as new services, design patterns, and product features become available. Determine which of these could improve performance or increase the efficiency of the workload through evaluation, internal discussion, or external analysis. Define a process to evaluate updates, new features, and services relevant to your workload. For example, build a proof of concept that uses new technologies or consult with an internal group. When trying new ideas or services, run performance tests to measure the impact that they have on the performance of the workload.

Implementation steps

- **Inventory your workload:** Inventory your workload software and architecture and identify components that need to be updated.
- **Identify update sources:** Identify news and update sources related to your workload components. As an example, you can subscribe to the What's New at AWS blog for the products that match your workload component. You can subscribe to the RSS feed or manage your email subscriptions.
- **Define an update schedule:** Define a schedule to evaluate new services and features for your workload.
  - You can use AWS Systems Manager Inventory to collect operating system (OS), application, and instance metadata from your Amazon EC2 instances and quickly understand which

instances are running the software and configurations required by your software policy and which instances need to be updated.

- **Assess the new update:** Understand how to update the components of your workload. Take advantage of agility in the cloud to quickly test how new features can improve your workload to gain performance efficiency.
- **Use automation:** Use automation for the update process to reduce the level of effort to deploy new features and limit errors caused by manual processes.
  - You can use CI/CD to automatically update AMIs, container images, and other artifacts related to your cloud application.
  - You can use tools such as AWS Systems Manager Patch Manager to automate the process of system updates, and schedule the activity using AWS Systems Manager Maintenance Windows.
- **Document the process:** Document your process for evaluating updates and new services. Provide your owners the time and space needed to research, test, experiment, and validate updates and new services. Refer back to the documented business requirements and KPIs to help prioritize which update will make a positive business impact.

Resources

Related documents:

- AWS Blog
- What's New with AWS
- Implementing up-to-date images with automated EC2 Image Builder pipelines

Related videos:

- AWS re:Inforce 2022 - Automating patch management and compliance using AWS
- All Things Patch: AWS Systems Manager | AWS Events

Related examples:

- Inventory and Patch Management
- One Observability Workshop

PERF05-BP07 Review metrics at regular intervals

As part of routine maintenance or in response to events or incidents, review which metrics are collected. Use these reviews to identify which metrics were essential in addressing issues and which additional metrics, if they were being tracked, could help identify, address, or prevent issues.

Common anti-patterns:

- You allow metrics to stay in an alarm state for an extended period of time.
- You create alarms that are not actionable by an automation system.

Benefits of establishing this best practice: Continually review metrics that are being collected to verify that they properly identify, address, or prevent issues. Metrics can also become stale if you let them stay in an alarm state for an extended period of time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Constantly improve metric collection and monitoring. As part of responding to incidents or events, evaluate which metrics were helpful in addressing the issue and which metrics could have helped that are not currently being tracked. Use this method to improve the quality of metrics you collect so that you can prevent, or more quickly resolve future incidents.

As part of responding to incidents or events, evaluate which metrics were helpful in addressing the issue and which metrics could have helped that are not currently being tracked. Use this to improve the quality of metrics you collect so that you can prevent or more quickly resolve future incidents.

Implementation steps

- **Define metrics:** Define critical performance metrics to monitor that are aligned to your workload objective, including metrics such as response time and resource utilization.
- **Establish baselines:** Set a baseline and desirable value for each metric. The baseline should provide reference points to identify deviation or anomalies.
- **Set up a cadence:** Set a cadence (like weekly or monthly) to review critical metrics.
- **Identify performance issues:** During each review, assess trends and deviation from the baseline values. Look for any performance bottlenecks or anomalies. For identified issues, conduct in-depth root cause analysis to understand the main reason behind the issue.

- **Identify corrective actions:** Use your analysis to identify corrective actions. This may include parameter tuning, fixing bugs, and scaling resources.
- **Document findings:** Document your findings, including identified issues, root causes, and corrective actions.
- **Iterate and improve:** Continually assess and improve the metrics review process. Use the lesson learned from previous review to enhance the process over time.

Resources

Related documents:

- CloudWatch Documentation
- Collect metrics and logs from Amazon EC2 Instances and on-premises servers with the CloudWatch Agent
- Query your metrics with CloudWatch Metrics Insights
- Monitoring, Logging, and Performance AWS Partner Network Partners
- X-Ray Documentation

Related videos:

- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS

Related examples:

- Creating a dashboard with Quick
- CloudWatch Dashboards

Cost optimization

The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point. You can find prescriptive guidance on implementation in the Cost Optimization Pillar whitepaper.

Best practice areas

- Practice Cloud Financial Management
- Expenditure and usage awareness
- Cost-effective resources
- Manage demand and supply resources
- Optimize over time

Practice Cloud Financial Management

Question

- COST 1. How do you implement cloud financial management?

COST 1. How do you implement cloud financial management?

Implementing Cloud Financial Management helps organizations realize business value and financial success as they optimize their cost and usage and scale on AWS.

Best practices

- COST01-BP01 Establish ownership of cost optimization
- COST01-BP02 Establish a partnership between finance and technology
- COST01-BP03 Establish cloud budgets and forecasts
- COST01-BP04 Implement cost awareness in your organizational processes
- COST01-BP05 Report and notify on cost optimization
- COST01-BP06 Monitor cost proactively
- COST01-BP07 Keep up-to-date with new service releases
- COST01-BP08 Create a cost-aware culture
- COST01-BP09 Quantify business value from cost optimization

COST01-BP01 Establish ownership of cost optimization

Create a team (Cloud Business Office, Cloud Center of Excellence, or FinOps team) that is responsible for establishing and maintaining cost awareness across your organization. The owner of cost optimization can be an individual or a team (requires people from finance, technology, and business teams) that understands the entire organization and cloud finance.

Level of risk exposed if this best practice is not established: High

Implementation guidance

This is the introduction of a Cloud Business Office (CBO) or Cloud Center of Excellence (CCOE) function or team that is responsible for establishing and maintaining a culture of cost awareness in cloud computing. This function can be an existing individual, a team within your organization, or a new team of key finance, technology, and organization stakeholders from across the organization.

The function (individual or team) prioritizes and spends the required percentage of their time on cost management and cost optimization activities. For a small organization, the function might spend a smaller percentage of time compared to a full-time function for a larger enterprise.

The function requires a multi-disciplinary approach, with capabilities in project management, data science, financial analysis, and software or infrastructure development. They can improve workload efficiency by running cost optimizations within three different ownerships:

- **Centralized:** Through designated teams such as FinOps team, Cloud Financial Management (CFM) team, Cloud Business Office (CBO), or Cloud Center of Excellence (CCoE), customers can design and implement governance mechanisms and drive best practices company-wide.
- **Decentralized:** Influencing technology teams to run cost optimizations.
- **Hybrid:** Combination of both centralized and decentralized teams can work together to run cost optimizations.

The function may be measured against their ability to run and deliver against cost optimization goals (for example, workload efficiency metrics).

You must secure executive sponsorship for this function, which is a key success factor. The sponsor is regarded as a champion for cost efficient cloud consumption, and provides escalation support for the team to ensure that cost optimization activities are treated with the level of priority defined by the organization. Otherwise, guidance can be ignored and cost saving opportunities will not be

prioritized. Together, the sponsor and team help your organization consume the cloud efficiently and deliver business value.

If you have the Business, Enterprise-On-Ramp or Enterprise support plan and need help building this team or function, reach out to your Cloud Financial Management (CFM) experts through your account team.

Implementation steps

- **Define key members:** All relevant parts of your organization must contribute and be interested in cost management. Common teams within organizations typically include: finance, application or product owners, management, and technical teams (DevOps). Some are engaged full time (finance or technical), while others are engaged periodically as required. Individuals or teams performing CFM need the following set of skills:
  - **Software development:** in the case where scripts and automation are being built out.
  - **Infrastructure engineering:** to deploy scripts, automate processes, and understand how services or resources are provisioned.
  - **Operations acumen:** CFM is about operating on the cloud efficiently by measuring, monitoring, modifying, planning, and scaling efficient use of the cloud.
- **Define goals and metrics:** The function needs to deliver value to the organization in different ways. These goals are defined and continually evolve as the organization evolves. Common activities include: creating and running education programs on cost optimization across the organization, developing organization-wide standards, such as monitoring and reporting for cost optimization, and setting workload goals on optimization. This function also needs to regularly report to the organization on their cost optimization capability.

  You can define value- or cost-based key performance indicators (KPIs). When you define the KPIs, you can calculate expected cost in terms of efficiency and expected business outcome. Value-based KPIs tie cost and usage metrics to business value drivers and help rationalize changes in AWS spend. The first step to deriving value-based KPIs is working together, cross-organizationally, to select and agree upon a standard set of KPIs.
- **Establish regular cadence:** The group (finance, technology and business teams) should come together regularly to review their goals and metrics. A typical cadence involves reviewing the state of the organization, reviewing any programs currently running, and reviewing overall financial and optimization metrics. Afterwards, key workloads are reported on in greater detail.

During these regular reviews, you can review workload efficiency (cost) and business outcome. For example, a 20% cost increase for a workload may align with increased customer usage. In this case, this 20% cost increase can be interpreted as an investment. These regular cadence calls can help teams to identify value KPIs that provide meaning to the entire organization.

Resources

Related documents:

- AWS CCOE Blog
- Creating Cloud Business Office
- CCOE - Cloud Center of Excellence

Related videos:

- Vanguard CCOE Success Story

Related examples:

- Using a Cloud Center of Excellence (CCOE) to Transform the Entire Enterprise
- Building a CCOE to transform the entire enterprise
- 7 Pitfalls to Avoid When Building CCOE

COST01-BP02 Establish a partnership between finance and technology

Involve finance and technology teams in cost and usage discussions at all stages of your cloud journey. Teams regularly meet and discuss topics such as organizational goals and targets, current state of cost and usage, and financial and accounting practices.

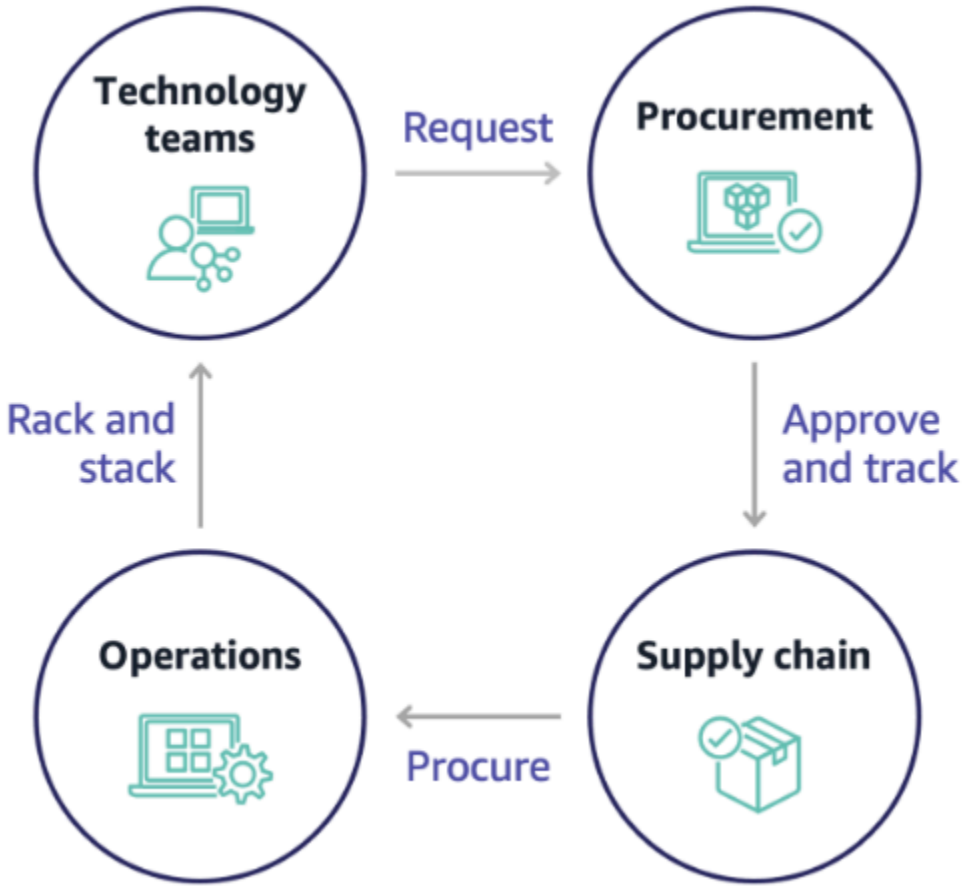
Level of risk exposed if this best practice is not established: High

Implementation guidance

Technology teams innovate faster in the cloud due to shortened approval, procurement, and infrastructure deployment cycles. This can be an adjustment for finance organizations previously used to running time-consuming and resource-intensive processes for procuring and deploying capital in data center and on-premises environments, and cost allocation only at project approval.

From a finance and procurement organization perspective, the process for capital budgeting, capital requests, approvals, procurement, and installing physical infrastructure is one that has been learned and standardized over decades:

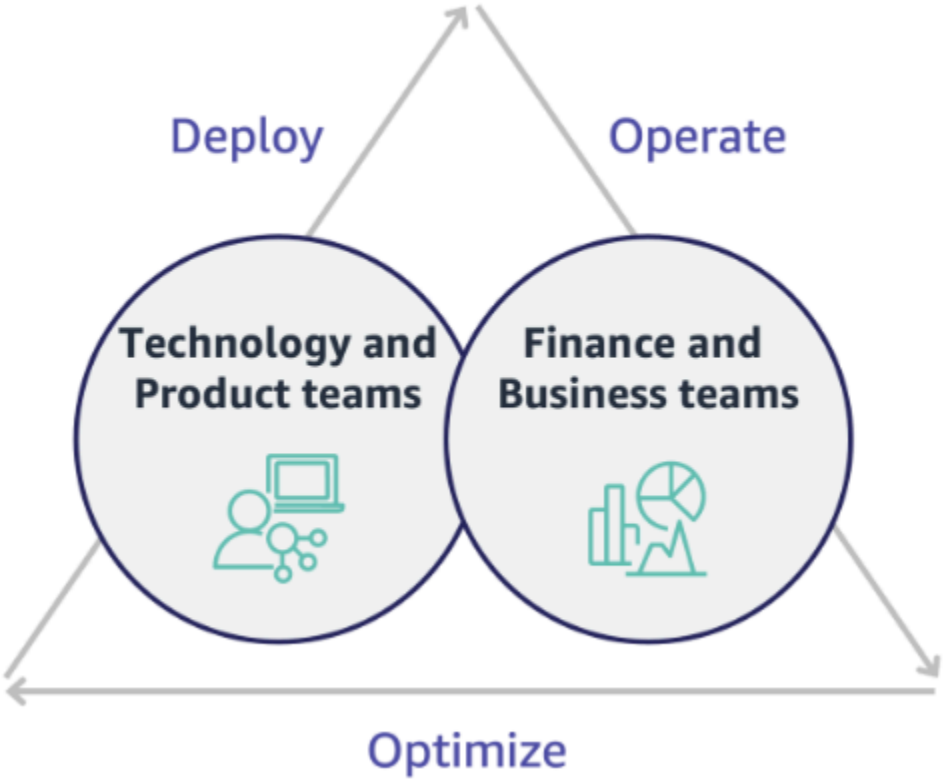
- Engineering or IT teams are typically the requesters
- Various finance teams act as approvers and procurers
- Operations teams rack, stack, and hand off ready-to-use infrastructure



With the adoption of cloud, infrastructure procurement and consumption are no longer beholden to a chain of dependencies. In the cloud model, technology and product teams are no longer just builders, but operators and owners of their products, responsible for most of the activities historically associated with finance and operations teams, including procurement and deployment.

All it really takes to provision cloud resources is an account, and the right set of permissions. This is also what reduces IT and finance risk; which means teams are always a just few clicks or API calls away from terminating idle or unnecessary cloud resources. This is also what allows technology

teams to innovate faster – the agility and ability to spin up and then tear down experiments. While the variable nature of cloud consumption may impact predictability from a capital budgeting and forecasting perspective, cloud provides organizations with the ability to reduce the cost of over-provisioning, as well as reduce the opportunity cost associated with conservative under-provisioning.



Establish a partnership between key finance and technology stakeholders to create a shared understanding of organizational goals and develop mechanisms to succeed financially in the variable spend model of cloud computing. Relevant teams within your organization must be involved in cost and usage discussions at all stages of your cloud journey, including:

- **Financial leads:** CFOs, financial controllers, financial planners, business analysts, procurement, sourcing, and accounts payable must understand the cloud model of consumption, purchasing options, and the monthly invoicing process. Finance needs to partner with technology teams to create and socialize an IT value story, helping business teams understand how technology spend is linked to business outcomes. This way, technology expenditures are viewed not as costs, but rather as investments. Due to the fundamental differences between the cloud (such as the rate of change in usage, pay as you go pricing, tiered pricing, pricing models, and detailed billing and usage information) compared to on-premises operation, it is essential that the finance

organization understands how cloud usage can impact business aspects including procurement processes, incentive tracking, cost allocation and financial statements.

- **Technology leads:** Technology leads (including product and application owners) must be aware of the financial requirements (for example, budget constraints) as well as business requirements (for example, service level agreements). This allows the workload to be implemented to achieve the desired goals of the organization.

The partnership of finance and technology provides the following benefits:

- Finance and technology teams have near real-time visibility into cost and usage.
- Finance and technology teams establish a standard operating procedure to handle cloud spend variance.
- Finance stakeholders act as strategic advisors with respect to how capital is used to purchase commitment discounts (for example, Reserved Instances or AWS Savings Plans), and how the cloud is used to grow the organization.
- Existing accounts payable and procurement processes are used with the cloud.
- Finance and technology teams collaborate on forecasting future AWS cost and usage to align and build organizational budgets.
- Better cross-organizational communication through a shared language, and common understanding of financial concepts.

Additional stakeholders within your organization that should be involved in cost and usage discussions include:

- **Business unit owners:** Business unit owners must understand the cloud business model so that they can provide direction to both the business units and the entire company. This cloud knowledge is critical when there is a need to forecast growth and workload usage, and when assessing longer-term purchasing options, such as Reserved Instances or Savings Plans.
- **Engineering team:** Establishing a partnership between finance and technology teams is essential for building a cost-aware culture that encourages engineers to take action on Cloud Financial Management (CFM). One of the common problems of CFM or finance operations practitioners and finance teams is getting engineers to understand the whole business on cloud, follow best practices, and take recommended actions.
- **Third parties:** If your organization uses third parties (for example, consultants or tools), ensure that they are aligned to your financial goals and can demonstrate both alignment through their

engagement models and a return on investment (ROI). Typically, third parties will contribute to reporting and analysis of any workloads that they manage, and they will provide cost analysis of any workloads that they design.

Implementing CFM and achieving success requires collaboration across finance, technology, and business teams, and a shift in how cloud spend is communicated and evaluated across the organization. Include engineering teams so that they can be part of these cost and usage discussions at all stages, and encourage them to follow best practices and take agreed-upon actions accordingly.

Implementation steps

- **Define key members:** Verify that all relevant members of your finance and technology teams participate in the partnership. Relevant finance members will be those having interaction with the cloud bill. This will typically be CFOs, financial controllers, financial planners, business analysts, procurement, and sourcing. Technology members will typically be product and application owners, technical managers and representatives from all teams that build on the cloud. Other members may include business unit owners, such as marketing, that will influence usage of products, and third parties such as consultants, to achieve alignment to your goals and mechanisms, and to assist with reporting.
- **Define topics for discussion:** Define the topics that are common across the teams, or will need a shared understanding. Follow cost from that time it is created, until the bill is paid. Note any members involved, and organizational processes that are required to be applied. Understand each step or process it goes through and the associated information, such as pricing models available, tiered pricing, discount models, budgeting, and financial requirements.
- **Establish regular cadence:** To create a finance and technology partnership, establish a regular communication cadence to create and maintain alignment. The group needs to come together regularly against their goals and metrics. A typical cadence involves reviewing the state of the organization, reviewing any programs currently running, and reviewing overall financial and optimization metrics. Then key workloads are reported on in greater detail.

Resources

Related documents:

- AWS News Blog

COST01-BP03 Establish cloud budgets and forecasts

Adjust existing organizational budgeting and forecasting processes to be compatible with the highly variable nature of cloud costs and usage. Processes must be dynamic, using trend-based or business driver-based algorithms or a combination of both.

Level of risk exposed if this best practice is not established: High

Implementation guidance

In traditional on-premises IT setups, customers often face the challenge of planning for fixed costs that change only occasionally, typically with the purchase of new IT hardware and services to meet peak demand. In contrast, AWS Cloud adopts a different approach, where customers pay for the resources they use as dictated by their actual IT and business needs. In the cloud environment, demand can fluctuate on a monthly, daily, or even hourly basis.

Using the cloud brings efficiency, speed, and agility, which results in a highly-variable cost and usage pattern. Costs can decrease or sometimes increase in response to greater workload efficiency or the deployment of new workloads and features. As workloads scale to serve an expanding customer base, cloud usage and costs correspondingly rise due to the increased accessibility of resources. This flexibility in cloud services extends to the costs and forecasts, which creates a degree of elasticity.

It's essential to align closely with these shifting business needs and demand drivers, and aim for the most accurate planning possible. Traditional organizational budget processes need to adapt to accommodate this variability.

Consider cost modeling while you forecast the cost for new workloads. Cost modeling creates a baseline understanding of expected cloud costs, which helps you perform total cost of ownership (TCO), return on investment (ROI), and other financial analysis, set targets and expectations with stakeholders, and identify opportunities for cost optimization.

Your organization should understand the cost definitions and accepted groupings. The level of detail at which you forecast can vary based on your organization's structure and internal workflows. Select a granularity that suits your specific requirements and organizational setup. It is important to understand at what level the forecast is performed:

- **Management account or AWS Organizations level:** The management account is the account that you use to create AWS Organizations. Organizations have one management account by default.

- **Linked or member account:** An account in Organizations is a standard AWS account that contains your AWS resources and the identities that can access those resources.
- **Environment:** An environment is a collection of AWS resources that runs an application version. An environment can be made with multiple linked or member accounts.
- **Project:** A project is a combination of set objectives or tasks to be accomplished within a fixed period. It is important to consider the project lifecycle during your forecast.
- **AWS services:** Groups or categories such as compute or storage services where you can group AWS services for your forecast.
- **Custom grouping:** You can create custom groups based on your organization's needs, such as business units, cost centers, teams, cost allocation tags, cost categories, linked accounts, or combination of these.

Identify the business drivers that can impact your usage cost, and forecast for each of them separately to calculate expected usage in advance. Some of the drivers might be linked to IT and product teams within the organization. Other business drivers, such as marketing events, promotions, geographic expansions, mergers, and acquisitions, are known by your sales, marketing, and business leaders, and it's important to collaborate and account for all those demand drivers as well.

You can use AWS Cost Explorer for trend-based forecasting in a defined future time range based on your past spend. AWS Cost Explorer's forecasting engine segments your historical data based on charge types (for example, Reserved Instances) and uses a combination of machine learning and rule-based models to predict spend across all charge types individually.

Once you've established your forecast process and built models, you can use AWS Budgets to set custom budgets at a granular level by specifying the time period, recurrence, or amount (fixed or variable) and add filters such as service, AWS Region, and tags. The budget is usually prepared for a single year and remains fixed, which requires strict adherence from everyone involved. In contrast, forecasts are more flexible, which allows for readjustments throughout the year and provides dynamic projections over a period of one, two, or three years. Both budgets and forecasts play a crucial role when you establish financial expectations among various technology and business stakeholders. Accurate forecasts and implementation also provides accountability to stakeholders who are directly responsible for provisioning cost in the first place, and it can also raise their overall cost awareness.

To stay informed on the performance of your existing budgets, you can create and schedule AWS Budgets reports to email you and your stakeholders on a regular cadence. You can also create AWS

Budgets alerts based on actual costs, which are reactive in nature, or on forecasted costs, which provides time to implement mitigations against potential cost overruns. You can be alerted when your cost or usage actually exceeds a certain level or if they are forecasted to exceed your budgeted amount.

Adjust existing budget and forecast processes to be more dynamic using trend-based algorithms (with historical costs as inputs) and driver-based algorithms (for example, new product launches, Regional expansion, or new environments for workloads), which are ideal for a dynamic and variable spending environment. Once you've determined your trend-based forecast using Cost Explorer or any other tools, use the AWS Pricing Calculator to estimate your AWS use case and future costs based on the expected usage (traffic, requests-per-second, or required Amazon EC2 instances).

Track the accuracy of that forecast, as budgets should be set based on these forecast calculations and estimations. Monitor the accuracy and effectiveness of the integrated cloud cost forecasts. Regularly review actual spend compared to your forecast, and adjust as needed to improve forecast precision. Track forecast variance, and perform root cause analysis on reported variance to act and adjust forecasts.

As mentioned in the COST01-BP02 Establish a partnership between finance and technology, it is important to foster a partnership and cadence between IT, finance, and other stakeholders to verify that they are all using the same tools or processes for consistency. In cases where budgets may need to change, increase cadence touch points to react to those changes more quickly.

Implementation steps

- **Define the cost language within the organization:** Create a common AWS cost language within the Organization with multiple dimension and grouping. Make sure stakeholders understand forecast granularity, pricing models, and the level of your cost forecasts.
- **Analyze trend-based forecasts:** Use trend-based forecast tools such as AWS Cost Explorer and Amazon Forecast. Analyze your usage cost on multiple dimensions like service, account, tags, and cost categories.
- **Analyze driver-based forecasts:** Identify the impact of business drivers on your cloud usage, and forecast for each of them separately to calculate expected usage cost in advance. Work closely with business unit owners and stakeholders to understand the impact on new drivers, and calculate expected cost changes to define accurate budgets.
- **Update existing forecast and budget processes:** Based on adopted forecast methods such as trend-based, business driver-based, or a combination of both forecasting methods, define

your forecast and budget processes. Budgets should be calculated, realistic, and based on your forecasts.

- **Configure alerts and notifications:** Use AWS Budgets alerts and cost anomaly detection to get alerts and notifications.
- **Perform regular reviews with key stakeholders:** For example, align on changes in business direction and usage with stakeholders in IT, finance, platform teams, and other areas of the business.

Resources

Related documents:

- AWS Cost Explorer
- AWS Cost and Usage Report
- Forecasting with Cost Explorer
- Quick Forecasting
- AWS Budgets

Related videos:

- How can I use AWS Budgets to track my spending and usage
- AWS Cost Optimization Series: AWS Budgets

Related examples:

- Understand and build driver-based forecasting
- How to establish and drive a forecasting culture
- How to improve your cloud cost forecasting
- Using the right tools for your cloud cost forecasting

COST01-BP04 Implement cost awareness in your organizational processes

Implement cost awareness, create transparency, and accountability of costs into new or existing processes that impact usage, and leverage existing processes for cost awareness. Implement cost awareness into employee training.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Cost awareness must be implemented in new and existing organizational processes. It is one of the foundational, prerequisite capabilities for other best practices. It is recommended to reuse and modify existing processes where possible — this minimizes the impact to agility and velocity. Report cloud costs to the technology teams and the decision makers in the business and finance teams to raise cost awareness, and establish efficiency key performance indicators (KPIs) for finance and business stakeholders. The following recommendations will help implement cost awareness in your workload:

- Verify that change management includes a cost measurement to quantify the financial impact of your changes. This helps proactively address cost-related concerns and highlight cost savings.
- Verify that cost optimization is a core component of your operating capabilities. For example, you can leverage existing incident management processes to investigate and identify root causes for cost and usage anomalies or cost overruns.
- Accelerate cost savings and business value realization through automation or tooling. When thinking about the cost of implementing, frame the conversation to include an return on investment (ROI) component to justify the investment of time or money.
- Allocate cloud costs by implementing showbacks or chargebacks for cloud spend, including spend on commitment-based purchase options, shared services and marketplace purchases to drive most cost-aware cloud consumption.
- Extend existing training and development programs to include cost-awareness training throughout your organization. It is recommended that this includes continuous training and certification. This will build an organization that is capable of self-managing cost and usage.
- Take advantage of free AWS native tools such as AWS Cost Anomaly Detection, AWS Budgets, and AWS Budgets Reports.

When organizations consistently adopt Cloud Financial Management (CFM) practices, those behaviours become ingrained in the way of working and decision-making. The result is a culture that is more cost-aware, from developers architecting a new born-in-the-cloud application, to finance managers analyzing the ROI on these new cloud investments.

Implementation steps

- **Identify relevant organizational processes:** Each organizational unit reviews their processes and identifies processes that impact cost and usage. Any processes that result in the creation or termination of a resource need to be included for review. Look for processes that can support cost awareness in your business, such as incident management and training.
- **Establish self-sustaining cost-aware culture:** Make sure all the relevant stakeholders align with cause-of-change and impact as a cost so that they understand cloud cost. This will allow your organization to establish a self-sustaining cost-aware culture of innovation.
- **Update processes with cost awareness:** Each process is modified to be made cost aware. The process may require additional pre-checks, such as assessing the impact of cost, or post-checks validating that the expected changes in cost and usage occurred. Supporting processes such as training and incident management can be extended to include items for cost and usage.

To get help, reach out to CFM experts through your Account team, or explore the resources and related documents below.

Resources

Related documents:

- AWS Cloud Financial Management

Related examples:

- Strategy for Efficient Cloud Cost Management
- Cost Control Blog Series #3: How to Handle Cost Shock
- A Beginner's Guide to AWS Cost Management

COST01-BP05 Report and notify on cost optimization

Set up cloud budgets and configure mechanisms to detect anomalies in usage. Configure related tools for cost and usage alerts against pre-defined targets and receive notifications when any usage exceeds those targets. Have regular meetings to analyze the cost-effectiveness of your workloads and promote cost awareness.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

You must regularly report on cost and usage optimization within your organization. You can implement dedicated sessions to discuss cost performance, or include cost optimization in your regular operational reporting cycles for your workloads. Use services and tools to monitor your cost performances regularly and implement cost savings opportunities.

View your cost and usage with multiple filters and granularity by using AWS Cost Explorer, which provides dashboards and reports such as costs by service or by account, daily costs, or marketplace costs. Track your progress of cost and usage against configured budgets with AWS Budgets Reports.

Use AWS Budgets to set custom budgets to track your costs and usage and respond quickly to alerts received from email or Amazon Simple Notification Service (Amazon SNS) notifications if you exceed your threshold. Set your preferred budget period to daily, monthly, quarterly, or annually, and create specific budget limits to stay informed on how actual or forecasted costs and usage progress toward your budget threshold. You can also set up alerts and actions against those alerts to run automatically or through an approval process when a budget target is exceeded.

Implement notifications on cost and usage to ensure that changes in cost and usage can be acted upon quickly if they are unexpected. AWS Cost Anomaly Detection allows you to reduce cost surprises and enhance control without slowing innovation. AWS Cost Anomaly Detection identifies anomalous spend and root causes, which helps to reduce the risk of billing surprises. With three simple steps, you can create your own contextualized monitor and receive alerts when any anomalous spend is detected.

You can also use Quick with AWS Cost and Usage Report (CUR) data, to provide highly customized reporting with more granular data. Quick allows you to schedule reports and receive periodic Cost Report emails for historical cost and usage or cost-saving opportunities. Check our Cost Intelligence Dashboard (CID) solution built on Quick, which gives you advanced visibility.

Use AWS Trusted Advisor, which provides guidance to verify whether provisioned resources are aligned with AWS best practices for cost optimization.

Check your Savings Plans recommendations through visual graphs against your granular cost and usage. Hourly graphs show On-Demand spend alongside the recommended Savings Plans commitment, providing insight into estimated savings, Savings Plans coverage, and Savings Plans utilization. This helps organizations to understand how their Savings Plans apply to each hour of spend without having to invest time and resources into building models to analyze their spend.

Periodically create reports containing a highlight of Savings Plans, Reserved Instances, and Amazon EC2 rightsizing recommendations from AWS Cost Explorer to start reducing the cost associated with steady-state workloads, idle, and underutilized resources. Identify and recoup spend associated with cloud waste for resources that are deployed. Cloud waste occurs when incorrectly-sized resources are created or different usage patterns are observed instead what is expected. Follow AWS best practices to reduce your waste or ask your account team and partner to help you to optimize and save your cloud costs.

Generate reports regularly for better purchasing options for your resources to drive down unit costs for your workloads. Purchasing options such as Savings Plans, Reserved Instances, or Amazon EC2 Spot Instances offer the deepest cost savings for fault-tolerant workloads and allow stakeholders (business owners, finance, and tech teams) to be part of these commitment discussions.

Share the reports that contain opportunities or new release announcements that may help you to reduce total cost of ownership (TCO) of the cloud. Adopt new services, Regions, features, solutions, or new ways to achieve further cost reductions.

Implementation steps

- **Configure AWS Budgets:** Configure AWS Budgets on all accounts for your workload. Set a budget for the overall account spend, and a budget for the workload by using tags.
  - Well-Architected Labs: Cost and Governance Usage
- **Report on cost optimization:** Set up a regular cycle to discuss and analyze the efficiency of the workload. Using the metrics established, report on the metrics achieved and the cost of achieving them. Identify and fix any negative trends, as well as positive trends that you can promote across your organization. Reporting should involve representatives from the application teams and owners, finance, and key decision makers with respect to cloud expenditure.

Resources

Related documents:

- AWS Cost Explorer
- AWS Trusted Advisor
- AWS Budgets
- AWS Cost and Usage Report

- AWS Budgets Best Practices
- Amazon S3 Analytics

Related examples:

- Key ways to start optimizing your AWS cloud costs

COST01-BP06 Monitor cost proactively

Implement tooling and dashboards to monitor cost proactively for the workload. Regularly review the costs with configured tools or out of the box tools, do not just look at costs and categories when you receive notifications. Monitoring and analyzing costs proactively helps to identify positive trends and allows you to promote them throughout your organization.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

It is recommended to monitor cost and usage proactively within your organization, not just when there are exceptions or anomalies. Highly visible dashboards throughout your office or work environment ensure that key people have access to the information they need, and indicate the organization's focus on cost optimization. Visible dashboards allow you to actively promote successful outcomes and implement them throughout your organization.

Create a daily or frequent routine to use AWS Cost Explorer or any other dashboard such as Amazon Quick to see the costs and analyze proactively. Analyze AWS service usage and costs at the AWS account-level, workload-level, or specific AWS service-level with grouping and filtering, and validate whether they are expected or not. Use the hourly- and resource-level granularity and tags to filter and identify incurring costs for the top resources. You can also build your own reports with the Cost Intelligence Dashboard, an Amazon Quick solution built by AWS Solutions Architects, and compare your budgets with the actual cost and usage.

Implementation steps

- **Report on cost optimization:** Set up a regular cycle to discuss and analyze the efficiency of the workload. Using the metrics established, report on the metrics achieved and the cost of achieving them. Identify and fix any negative trends, and identify positive trends to promote across your organization. Reporting should involve representatives from the application teams and owners, finance, and management.

- **Create and activate daily granularity AWS Budgets for the cost and usage to take timely actions to prevent any potential cost overruns:** AWS Budgets allow you to configure alert notifications, so you stay informed if any of your budget types fall out of your pre-configured thresholds. The best way to leverage AWS Budgets is to set your expected cost and usage as your limits, so that anything above your budgets can be considered overspend.
- **Create AWS Cost Anomaly Detection for cost monitor:** AWS Cost Anomaly Detection uses advanced Machine Learning technology to identify anomalous spend and root causes, so you can quickly take action. It allows you to configure cost monitors that define spend segments you want to evaluate (for example, individual AWS services, member accounts, cost allocation tags, and cost categories), and lets you set when, where, and how you receive your alert notifications. For each monitor, attach multiple alert subscriptions for business owners and technology teams, including a name, a cost impact threshold, and alerting frequency (individual alerts, daily summary, weekly summary) for each subscription.
- **Use AWS Cost Explorer or integrate your AWS Cost and Usage Report (CUR) data with Amazon Quick dashboards to visualize your organization's costs:** AWS Cost Explorer has an easy-to-use interface that lets you visualize, understand, and manage your AWS costs and usage over time. The Cost Intelligence Dashboard is a customizable and accessible dashboard to help create the foundation of your own cost management and optimization tool.

Resources

Related documents:

- AWS Budgets
- AWS Cost Explorer
- Daily Cost and Usage Budgets
- AWS Cost Anomaly Detection

Related examples:

- AWS Cost Anomaly Detection Alert with Slack

COST01-BP07 Keep up-to-date with new service releases

Consult regularly with experts or AWS Partners to consider which services and features provide lower cost. Review AWS blogs and other information sources.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS is constantly adding new capabilities so you can leverage the latest technologies to experiment and innovate more quickly. You may be able to implement new AWS services and features to increase cost efficiency in your workload. Regularly review AWS Cost Management, the AWS News Blog, the AWS Cost Management blog, and What's New with AWS for information on new service and feature releases. What's New posts provide a brief overview of all AWS service, feature, and Region expansion announcements as they are released.

Implementation steps

- **Subscribe to blogs:** Go to the AWS blogs pages and subscribe to the What's New Blog and other relevant blogs. You can sign up on the communication preference page with your email address.
- **Subscribe to AWS News:** Regularly review the AWS News Blog and What's New with AWS for information on new service and feature releases. Subscribe to the RSS feed, or with your email to follow announcements and releases.
- **Follow AWS Price Reductions:** Regular price cuts on all our services has been a standard way for AWS to pass on the economic efficiencies to our customers gained from our scale. As of September 20, 2023, AWS has reduced prices 134 times since 2006. If you have any pending business decisions due to price concerns, you can review them again after price reductions and new service integrations. You can learn about the previous price reductions efforts, including Amazon Elastic Compute Cloud (Amazon EC2) instances, in the price-reduction category of the AWS News Blog.
- **AWS events and meetups:** Attend your local AWS summit, and any local meetups with other organizations from your local area. If you cannot attend in person, try to attend virtual events to hear more from AWS experts and other customers' business cases.
- **Meet with your account team:** Schedule a regular cadence with your account team, meet with them and discuss industry trends and AWS services. Speak with your account manager, Solutions Architect, and support team.

Resources

Related documents:

- AWS Cost Management

- What's New with AWS
- AWS News Blog

Related examples:

- Amazon EC2 – 15 Years of Optimizing and Saving Your IT Costs
- AWS News Blog - Price Reduction

COST01-BP08 Create a cost-aware culture

Implement changes or programs across your organization to create a cost-aware culture. It is recommended to start small, then as your capabilities increase and your organization's use of the cloud increases, implement large and wide ranging programs.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

A cost-aware culture allows you to scale cost optimization and Cloud Financial Management (financial operations, cloud center of excellence, cloud operations teams, and so on) through best practices that are performed in an organic and decentralized manner across your organization. Cost awareness allows you to create high levels of capability across your organization with minimal effort, compared to a strict top-down, centralized approach.

Creating cost awareness in cloud computing, especially for primary cost drivers in cloud computing, allows teams to understand expected outcomes of any changes in cost perspective. Teams who access the cloud environments should be aware of pricing models and the difference between traditional on-premesis datacenters and cloud computing.

The main benefit of a cost-aware culture is that technology teams optimize costs proactively and continually (for example, they are considered a non-functional requirement when architecting new workloads, or making changes to existing workloads) rather than performing reactive cost optimizations as needed.

Small changes in culture can have large impacts on the efficiency of your current and future workloads. Examples of this include:

- Giving visibility and creating awareness in engineering teams to understand what they do, and what they impact in terms of cost.

- Gamifying cost and usage across your organization. This can be done through a publicly visible dashboard, or a report that compares normalized costs and usage across teams (for example, cost-per-workload and cost-per-transaction).
- Recognizing cost efficiency. Reward voluntary or unsolicited cost optimization accomplishments publicly or privately, and learn from mistakes to avoid repeating them in the future.
- Creating top-down organizational requirements for workloads to run at pre-defined budgets.
- Questioning business requirements of changes, and the cost impact of requested changes to the architecture infrastructure or workload configuration to make sure you pay only what you need.
- Making sure the change planner is aware of expected changes that have a cost impact, and that they are confirmed by the stakeholders to deliver business outcomes cost-effectively.

Implementation steps

- **Report cloud costs to technology teams:** To raise cost awareness, and establish efficiency KPIs for finance and business stakeholders.
- **Inform stakeholders or team members about planned changes:** Create an agenda item to discuss planned changes and the cost-benefit impact on the workload during weekly change meetings.
- **Meet with your account team:** Establish a regular meeting cadence with your account team, and discuss industry trends and AWS services. Speak with your account manager, architect, and support team.
- **Share success stories:** Share success stories about cost reduction for any workload, AWS account, or organization to create a positive attitude and encouragement around cost optimization.
- **Training:** Ensure technical teams or team members are trained for awareness of resource costs on AWS Cloud.
- **AWS events and meetups:** Attend local AWS summits, and any local meetups with other organizations from your local area.
- **Subscribe to blogs:** Go to the AWS blogs pages and subscribe to the What's New Blog and other relevant blogs to follow new releases, implementations, examples, and changes shared by AWS.

Resources

Related documents:

- AWS Blog

- AWS Cost Management
- AWS News Blog

Related examples:

- AWS Cloud Financial Management

COST01-BP09 Quantify business value from cost optimization

Quantifying business value from cost optimization allows you to understand the entire set of benefits to your organization. Because cost optimization is a necessary investment, quantifying business value allows you to explain the return on investment to stakeholders. Quantifying business value can help you gain more buy-in from stakeholders on future cost optimization investments, and provides a framework to measure the outcomes for your organization's cost optimization activities.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Quantifying the business value means measuring the benefits that businesses gain from the actions and decisions they take. Business value can be tangible (like reduced expenses or increased profits) or intangible (like improved brand reputation or increased customer satisfaction).

To quantify business value from cost optimization means determining how much value or benefit you're getting from your efforts to spend more efficiently. For example, if a company spends $100,000 to deploy a workload on AWS and later optimizes it, the new cost becomes only $80,000 without sacrificing the quality or output. In this scenario, the quantified business value from cost optimization would be a savings of $20,000. But beyond just savings, the business might also quantify value in terms of faster delivery times, improved customer satisfaction, or other metrics that result from the cost optimization efforts. Stakeholders need to make decisions about the potential value of cost optimization, the cost of optimizing the workload, and return value.

In addition to reporting savings from cost optimization, it is recommended that you quantify the additional value delivered. Cost optimization benefits are typically quantified in terms of lower costs per business outcome. For example, you can quantify Amazon Elastic Compute Cloud(Amazon EC2) cost savings when you purchase Savings Plans, which reduce cost and maintain workload output levels. You can quantify cost reductions in AWS spending when idle Amazon EC2 instances are removed, or unattached Amazon Elastic Block Store (Amazon EBS) volumes are deleted.

The benefits from cost optimization, however, go above and beyond cost reduction or avoidance. Consider capturing additional data to measure efficiency improvements and business value.

Implementation steps

- **Evaluate business benefits:** This is the process of analyzing and adjusting AWS Cloud cost in ways that maximize the benefit received from each dollar spent. Instead of focusing on cost reduction without business value, consider business benefits and return on investments for cost optimization, which may bring more value out of the money you spend. It's about spending wisely and making investments and expenditures in areas that yield the best return.
- **Analyze forecasting AWS costs:** Forecasting helps finance stakeholders set expectations with other internal and external organization stakeholders, and can improve your organization's financial predictability. AWS Cost Explorer can be used to perform forecasting for your cost and usage.

Resources

Related documents:

- AWS Cloud Economics
- AWS Blog
- AWS Cost Management
- AWS News Blog
- Well-Architected Reliability Pillar whitepaper
- AWS Cost Explorer

Related videos:

- Unlock Business Value with Windows on AWS

Related examples:

- Measuring and Maximizing the Business Value of Customer 360
- The Business Value of Adopting Amazon Web Services Managed Databases
- The Business Value of Amazon Web Services for Independent Software Vendors
- Business Value of Cloud Modernization

- The Business Value of Migration to Amazon Web Services

Expenditure and usage awareness

Questions

- COST 2. How do you govern usage?
- COST 3. How do you monitor your cost and usage?
- COST 4. How do you decommission resources?

COST 2. How do you govern usage?

Establish policies and mechanisms to verify that appropriate costs are incurred while objectives are achieved. By employing a checks-and-balances approach, you can innovate without overspending.

Best practices

- COST02-BP01 Develop policies based on your organization requirements
- COST02-BP02 Implement goals and targets
- COST02-BP03 Implement an account structure
- COST02-BP04 Implement groups and roles
- COST02-BP05 Implement cost controls
- COST02-BP06 Track project lifecycle

COST02-BP01 Develop policies based on your organization requirements

Develop policies that define how resources are managed by your organization and inspect them periodically. Policies should cover the cost aspects of resources and workloads, including creation, modification, and decommissioning over a resource's lifetime.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understanding your organization's costs and drivers is critical for managing your cost and usage effectively and identifying cost reduction opportunities. Organizations typically operate multiple workloads run by multiple teams. These teams can be in different organization units, each with

its own revenue stream. The capability to attribute resource costs to the workloads, individual organization, or product owners drives efficient usage behaviour and helps reduce waste. Accurate cost and usage monitoring helps you understand how optimized a workload is, as well as how profitable organization units and products are. This knowledge allows for more informed decision making about where to allocate resources within your organization. Awareness of usage at all levels in the organization is key to driving change, as change in usage drives changes in cost. Consider taking a multi-faceted approach to becoming aware of your usage and expenditures.

The first step in performing governance is to use your organization's requirements to develop policies for your cloud usage. These policies define how your organization uses the cloud and how resources are managed. Policies should cover all aspects of resources and workloads that relate to cost or usage, including creation, modification, and decommissioning over a resource's lifetime. Verify that policies and procedures are followed and implemented for any change in a cloud environment. During your IT change management meetings, raise questions to find out the cost impact of planned changes, whether increasing or decreasing, the business justification, and the expected outcome.

Policies should be simple so that they are easily understood and can be implemented effectively throughout the organization. Policies also need to be easy to follow and interpret (so they are used) and specific (no misinterpretation between teams). Moreover, they need to be inspected periodically (like our mechanisms) and updated as customers business conditions or priorities change, which would make the policy outdated.

Start with broad, high-level policies, such as which geographic Region to use or times of the day that resources should be running. Gradually refine the policies for the various organizational units and workloads. Common policies include which services and features can be used (for example, lower performance storage in test and development environments), which types of resources can be used by different groups (for example, the largest size of resource in a development account is medium) and how long these resources will be in use (whether temporary, short term, or for a specific period of time).

Policy example

The following is a sample policy you can review to create your own cloud governance policies, which focus on cost optimization. Make sure you adjust policy based on your organization's requirements and your stakeholders' requests.

- **Policy name:** Define a clear policy name, such as Resource Optimization and Cost Reduction Policy.

- **Purpose:** Explain why this policy should be used and what is the expected outcome. The objective of this policy is to verify that there is a minimum cost required to deploy and run the desired workload to meet business requirements.
- **Scope:** Clearly define who should use this policy and when it should be used, such as DevOps X Team to use this policy in us-east customers for X environment (production or non-production).

Policy statement

1. Select us-east-1or multiple us-east regions based on your workload's environment and business requirement (development, user acceptance testing, pre-production, or production).
2. Schedule Amazon EC2 and Amazon RDS instances to run between six in the morning and eight at night (Eastern Standard Time (EST)).
3. Stop all unused Amazon EC2 instances after eight hours and unused Amazon RDS instances after 24 hours of inactivity.
4. Terminate all unused Amazon EC2 instances after 24 hours of inactivity in non-production environments. Remind Amazon EC2 instance owner (based on tags) to review their stopped Amazon EC2 instances in production and inform them that their Amazon EC2 instances will be terminated within 72 hours if they are not in use.
5. Use generic instance family and size such as m5.large and then resize the instance based on CPU and memory utilization using AWS Compute Optimizer.
6. Prioritize using auto scaling to dynamically adjust the number of running instances based on traffic.
7. Use spot instances for non-critical workloads.
8. Review capacity requirements to commit saving plans or reserved instances for predictable workloads and inform Cloud Financial Management Team.
9. Use Amazon S3 lifecycle policies to move infrequently accessed data to cheaper storage tiers. If no retention policy defined, use Amazon S3 Intelligent Tiering to move objects to archived tier automatically.
10. Monitor resource utilization and set alarms to trigger scaling events using Amazon CloudWatch.
11. For each AWS account, use AWS Budgets to set cost and usage budgets for your account based on cost center and business units.
12. Using AWS Budgets to set cost and usage budgets for your account can help you stay on top of your spending and avoid unexpected bills, allowing you to better control your costs.

Procedure: Provide detailed procedures for implementing this policy or refer to other documents that describe how to implement each policy statement. This section should provide step-by-step instructions for carrying out the policy requirements.

To implement this policy, you can use various third-party tools or AWS Config rules to check for compliance with the policy statement and trigger automated remediation actions using AWS Lambda functions. You can also use AWS Organizations to enforce the policy. Additionally, you should regularly review your resource usage and adjust the policy as necessary to verify that it continues to meet your business needs.

Implementation steps

- **Meet with stakeholders:** To develop policies, ask stakeholders (cloud business office, engineers, or functional decision makers for policy enforcement) within your organization to specify their requirements and document them. Take an iterative approach by starting broadly and continually refine down to the smallest units at each step. Team members include those with direct interest in the workload, such as organization units or application owners, as well as supporting groups, such as security and finance teams.
- **Get confirmation:** Make sure teams agree on policies who can access and deploy to the AWS Cloud. Make sure they follow your organization's policies and confirm that their resource creations align with the agreed policies and procedures.
- **Create onboarding training sessions:** Ask new organization members to complete onboarding training courses to create cost awareness and organization requirements. They may assume different policies from their previous experience or not think of them at all.
- **Define locations for your workload:** Define where your workload operates, including the country and the area within the country. This information is used for mapping to AWS Regions and Availability Zones.
- **Define and group services and resources:** Define the services that the workloads require. For each service, specify the types, the size, and the number of resources required. Define groups for the resources by function, such as application servers or database storage. Resources can belong to multiple groups.
- **Define and group the users by function:** Define the users that interact with the workload, focusing on what they do and how they use the workload, not on who they are or their position in the organization. Group similar users or functions together. You can use the AWS managed policies as a guide.
- **Define the actions:** Using the locations, resources, and users identified previously, define the actions that are required by each to achieve the workload outcomes over its life time

(development, operation, and decommission). Identify the actions based on the groups, not the individual elements in the groups, in each location. Start broadly with read or write, then refine down to specific actions to each service.

- **Define the review period:** Workloads and organizational requirements can change over time. Define the workload review schedule to ensure it remains aligned with organizational priorities.
- **Document the policies:** Verify the policies that have been defined are accessible as required by your organization. These policies are used to implement, maintain, and audit access of your environments.

Resources

Related documents:

- Change Management in the Cloud
- AWS Managed Policies for Job Functions
- AWS multiple account billing strategy
- Actions, Resources, and Condition Keys for AWS Services
- AWS Management and Governance
- Control access to AWS Regions using IAM policies
- Global Infrastructures Regions and AZs

Related videos:

- AWS Management and Governance at Scale

COST02-BP02 Implement goals and targets

Implement both cost and usage goals and targets for your workload. Goals provide direction to your organization on expected outcomes, and targets provide specific measurable outcomes to be achieved for your workloads.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Develop cost and usage goals and targets for your organization. As a growing organization on AWS, it is important to set and track goals for cost optimization. These goals or key performance

indicators (KPIs) can include things like percent of spend on-demand or adoption of certain optimized services such as AWS Graviton instances or gp3 EBS volume types. Set measurable and achievable goals to help you measure efficiency improvements, which is important for your business operations. Goals provide guidance and direction to your organization on expected outcomes.

Targets provide specific, measurable outcomes to be achieved. In short, a goal is the direction you want to go, and a target is how far in that direction and when that goal should be achieved (use guidance of specific, measurable, assignable, realistic and timely, or SMART). An example of a goal is that platform usage should increase significantly, with only a minor (non-linear) increase in cost. An example target is a 20% increase in platform usage, with less than a five percent increase in costs. Another common goal is that workloads need to be more efficient every six months. The accompanying target would be that the cost per business metrics needs to decrease by five percent every six months. Use the right metrics, and set calculated KPIs for your organization. You can start with basic KPIs and evolve later based on business needs.

A goal for cost optimization is to increase workload efficiency, which corresponds to a decrease in the cost per business outcome of the workload over time. Implement this goal for all workloads, and set a target like a five percent increase in efficiency every six months to a year. In the cloud, you can achieve this through the establishment of capability in cost optimization, as well as new service and feature releases.

Targets are the quantifiable benchmarks you want to reach to meet your goals and benchmarks compare your actual results against a target. Establish benchmarks with KPIs for the cost per unit of compute services (such as Spot adoption, Graviton adoption, latest instance types, and On-Demands coverage), storage services (such as EBS GP3 adoption, obsolete EBS snapshots, and Amazon S3 standard storage), or database service usage (such as RDS open-source engines, Graviton adoption, and On-demand coverage). These benchmarks and KPIs can help you verify that you use AWS services in the most cost-effective manner.

The following table provides a list of standard AWS metrics for reference. Each organization can have different target values for these KPIs.

Category	KPI (%)	Description
Compute	EC2 usage Coverage	EC2 instances (in cost or hours) using SP+RI+Spot

Category	KPI (%)	Description
		compared to total (in cost or hours) of EC2 instances
Compute	Compute SP/RI utilization	Utilized SP or RI hours compared to total available SP or RI hours
Compute	EC2/Hour cost	EC2 cost divided by the number of EC2 instances running in that hour
Compute	vCPU cost	Cost per vCPU for all instances
Compute	Latest Instance Generation	Percentage of instances on Graviton (or other modern generation instance types)
Database	RDS coverage	RDS instances (in cost or hours) using RI compared to total (in cost or hours) of RDS instances
Database	RDS utilization	Utilized RI hours compared to total available RI hours
Database	RDS uptime	RDS cost divided by the number of RDS instances running in that hour
Database	Latest Instance Generation	Percentage of instances on Graviton (or other modern instance types)

Category	KPI (%)	Description
Storage	Storage utilization	Optimized storage cost (for example Glacier, deep archive, or Infrequent Access) divided by total storage cost
Tagging	Untagged resources	Cost Explorer: 1. Filter out credits, discounts , taxes, refunds, marketplace, and copy the latest monthly cost 2. Select **Show only untagged resources** in Cost Explorer 3. Divide the amount in **untagged resources** with your monthly cost.

Using this table, include target or benchmark values, which should be calculated based on your organizational goals. You need to measure certain metrics for your business and understand business outcome for that workload to define accurate and realistic KPIs. When you evaluate performance metrics within an organization, distinguish between different types of metrics that serve distinct purposes. These metrics primarily measure the performance and efficiency of the technical infrastructure rather than directly the overall business impact. For instance, they might track server response times, network latency, or system uptime. These metrics are crucial to assess how well the infrastructure supports the organization's technical operations. However, they don't provide direct insight into broader business objectives like customer satisfaction, revenue growth, or market share. To gain a comprehensive understanding of business performance, complement these efficiency metrics with strategic business metrics that directly correlate with business outcomes.

Establish near real-time visibility over your KPIs and related savings opportunities and track your progress over time. To get started with the definition and tracking of KPI goals, we recommend the KPI dashboard from Cloud Intelligence Dashboards (CID). Based on the data from Cost and Usage

Report (CUR), the KPI dashboard provides a series of recommended cost optimization KPIs, with the ability to set custom goals and track progress over time.

If you have other solutions to set and track KPI goals, make sure these methods are adopted by all cloud financial management stakeholders in your organization.

Implementation steps

- **Define expected usage levels:** To begin, focus on usage levels. Engage with the application owners, marketing, and greater business teams to understand what the expected usage levels are for the workload. How might customer demand change over time, and what can change due to seasonal increases or marketing campaigns?
- **Define workload resourcing and costs:** With usage levels defined, quantify the changes in workload resources required to meet those usage levels. You may need to increase the size or number of resources for a workload component, increase data transfer, or change workload components to a different service at a specific level. Specify the costs at each of these major points, and predict the change in cost when there is a change in usage.
- **Define business goals:** Take the output from the expected changes in usage and cost, combine this with expected changes in technology, or any programs that you are running, and develop goals for the workload. Goals must address usage and cost, as well as the relationship between the two. Goals must be simple, high-level, and help people understand what the business expects in terms of outcomes (such as making sure unused resources are kept below certain cost level). You don't need to define goals for each unused resource type or define costs that can cause losses in goals and targets. Verify that there are organizational programs (for example, capability building like training and education) if there are expected changes in cost without changes in usage.
- **Define targets:** For each of the defined goals, specify a measurable target. If the goal is to increase efficiency in the workload, the target should quantify the amount of improvement (typically in business outputs for each dollar spent) and when it should be delivered. For example, you could set a goal to minimize waste due to over-provisioning. With this goal, your target can be that waste due to compute over-provisioning in the first tier of production workloads should not exceed ten percent of tier compute cost. Additionally, a second target could be that waste due to compute over-provisioning in the second tier of production workloads should not exceed five percent of tier compute cost.

Resources

Related documents:

- AWS managed policies for job functions
- AWS multiple account billing strategy
- Control access to AWS Regions using IAM policies
- S.M.A.R.T. Goals
- How to track your cost optimization KPIs with the CID KPI Dashboard

Related videos:

- Well-Architected Labs: Goals and Targets (Level 100)

Related examples:

- What is a unit metric?
- Selecting a unit metric to support your business
- Unit metrics in practice – lessons learned
- How unit metrics help create alignment between business functions

COST02-BP03 Implement an account structure

Implement a structure of accounts that maps to your organization. This assists in allocating and managing costs throughout your organization.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS Organizations allows you to create multiple AWS accounts which can help you centrally govern your environment as you scale your workloads on AWS. You can model your organizational hierarchy by grouping AWS accounts in organizational unit (OU) structure and creating multiple AWS accounts under each OU. To create an account structure, you need to decide which of your AWS accounts will be the management account first. After that, you can create new AWS accounts or select existing accounts as member accounts based on your designed account structure by following management account best practices and member account best practices.

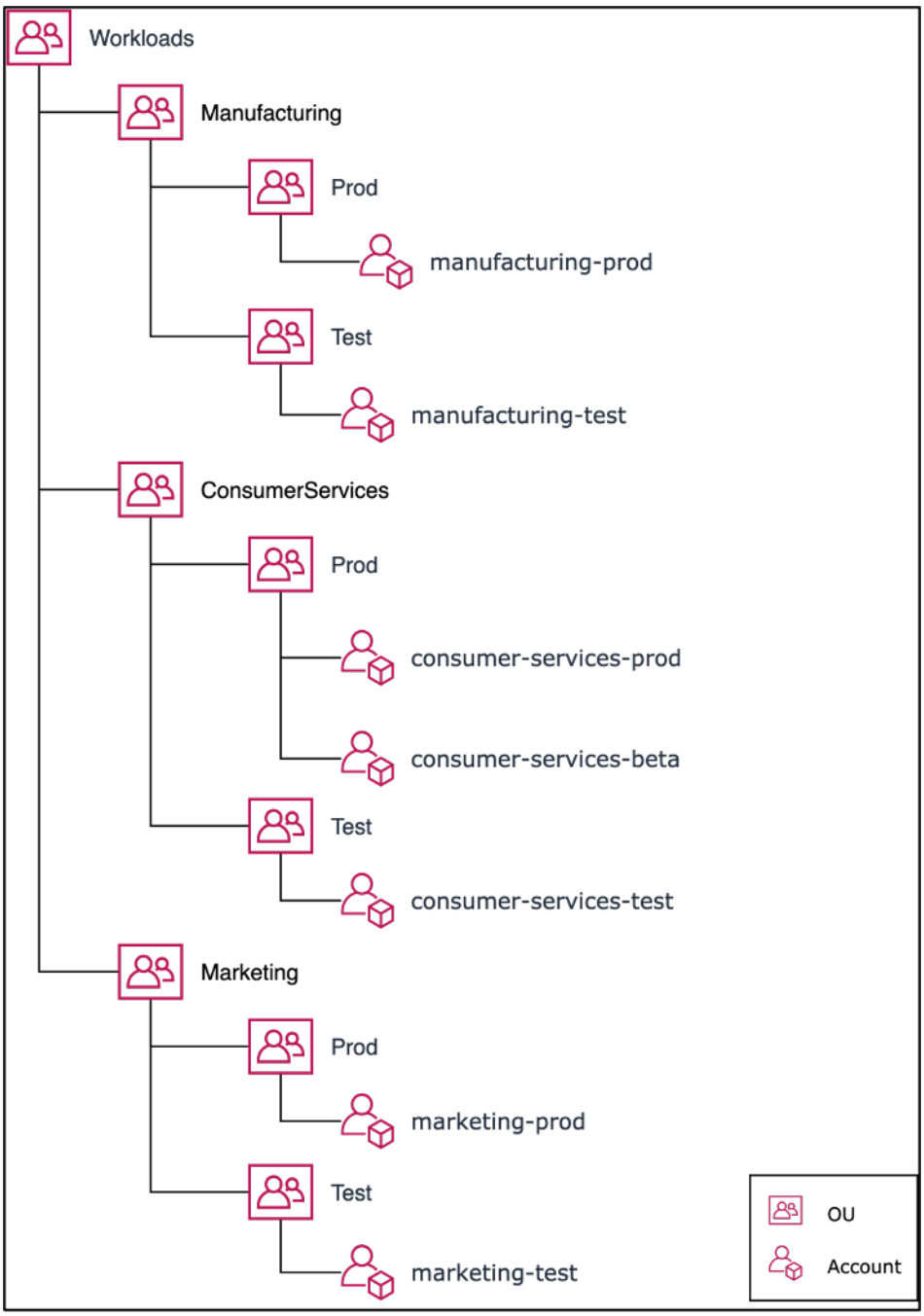
It is advised to always have at least one management account with one member account linked to it, regardless of your organization size or usage. All workload resources should reside only within member accounts and no resource should be created in management account. There is no one size fits all answer for how many AWS accounts you should have. Assess your current and future operational and cost models to ensure that the structure of your AWS accounts reflects your organization's goals. Some companies create multiple AWS accounts for business reasons, for example:

- Administrative or fiscal and billing isolation is required between organization units, cost centers, or specific workloads.
- AWS service limits are set to be specific to particular workloads.
- There is a requirement for isolation and separation between workloads and resources.

Within AWS Organizations, consolidated billing creates the construct between one or more member accounts and the management account. Member accounts allow you to isolate and distinguish your cost and usage by groups. A common practice is to have separate member accounts for each organization unit (such as finance, marketing, and sales), or for each environment lifecycle (such as development, testing and production), or for each workload (workload a, b, and c), and then aggregate these linked accounts using consolidated billing.

Consolidated billing allows you to consolidate payment for multiple member AWS accounts under a single management account, while still providing visibility for each linked account's activity. As costs and usage are aggregated in the management account, this allows you to maximize your service volume discounts, and maximize the use of your commitment discounts (Savings Plans and Reserved Instances) to achieve the highest discounts.

The following diagram shows how you can use AWS Organizations with organizational units (OU) to group multiple accounts, and place multiple AWS accounts under each OU. It is recommended to use OUs for various use cases and workloads which provides patterns for organizing accounts.



Example of grouping multiple AWS accounts under organizational units.

AWS Control Tower can quickly set up and configure multiple AWS accounts, ensuring that governance is aligned with your organization's requirements.

Implementation steps

- **Define separation requirements:** Requirements for separation are a combination of multiple factors, including security, reliability, and financial constructs. Work through each factor in order

and specify whether the workload or workload environment should be separate from other workloads. Security promotes adhesion to access and data requirements. Reliability manages limits so that environments and workloads do not impact others. Review the security and reliability pillars of the Well-Architected Framework periodically and follow the provided best practices. Financial constructs create strict financial separation (different cost center, workload ownerships and accountability). Common examples of separation are production and test workloads being run in separate accounts, or using a separate account so that the invoice and billing data can be provided to the individual business units or departments in the organization or stakeholder who owns the account.

- **Define grouping requirements:** Requirements for grouping do not override the separation requirements, but are used to assist management. Group together similar environments or workloads that do not require separation. An example of this is grouping multiple test or development environments from one or more workloads together.
- **Define account structure:** Using these separations and groupings, specify an account for each group and maintain separation requirements. These accounts are your member or linked accounts. By grouping these member accounts under a single management or payer account, you combine usage, which allows for greater volume discounts across all accounts, which provides a single bill for all accounts. It's possible to separate billing data and provide each member account with an individual view of their billing data. If a member account must not have its usage or billing data visible to any other account, or if a separate bill from AWS is required, define multiple management or payer accounts. In this case, each member account has its own management or payer account. Resources should always be placed in member or linked accounts. The management or payer accounts should only be used for management.

Resources

Related documents:

- Using Cost Allocation Tags
- AWS managed policies for job functions
- AWS multiple account billing strategy
- Control access to AWS Regions using IAM policies
- AWS Control Tower
- AWS Organizations
- Best practices for management accounts and member accounts

- Organizing Your AWS Environment Using Multiple Accounts
- Turning on shared reserved instances and Savings Plans discounts
- Consolidated billing
- Consolidated billing

Related examples:

- Splitting the CUR and Sharing Access

Related videos:

- Introducing AWS Organizations
- Set Up a Multi-Account AWS Environment that Uses Best Practices for AWS Organizations

Related examples:

- Defining an AWS Multi-Account Strategy for telecommunications companies
- Best Practices for Optimizing AWS accounts
- Best Practices for Organizational Units with AWS Organizations

COST02-BP04 Implement groups and roles

Implement groups and roles that align to your policies and control who can create, modify, or decommission instances and resources in each group. For example, implement development, test, and production groups. This applies to AWS services and third-party solutions.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

User roles and groups are fundamental building blocks in the design and implementation of secure and efficient systems. Roles and groups help organizations balance the need for control with the requirement for flexibility and productivity, ultimately supporting organizational objectives and user needs. As recommended in Identity and access management section of AWS Well-Architected Framework Security Pillar, you need robust identity management and permissions in place to provide access to the right resources for the right people under the right conditions. Users

receive only the access necessary to complete their tasks. This minimizes the risk associated with unauthorized access or misuse.

After you develop policies, you can create logical groups and user roles within your organization. This allows you to assign permissions, control usage, and help implement robust access control mechanisms, preventing unauthorized access to sensitive information. Begin with high-level groupings of people. Typically, this aligns with organizational units and job roles (for example, a systems administrator in the IT Department, financial controller, or business analysts). The groups categorize people that do similar tasks and need similar access. Roles define what a group must do. It is easier to manage permissions for groups and roles than for individual users. Roles and groups assign permissions consistently and systematically across all users, preventing errors and inconsistencies.

When a user's role changes, administrators can adjust access at the role or group level, rather than reconfiguring individual user accounts. For example, a systems administrator in IT requires access to create all resources, but an analytics team member only needs to create analytics resources.

Implementation steps

- **Implement groups:** Using the groups of users defined in your organizational policies, implement the corresponding groups, if necessary. For best practices on users, groups and authentication, see the Security Pillar of the AWS Well-Architected Framework.
- **Implement roles and policies:** Using the actions defined in your organizational policies, create the required roles and access policies. For best practices on roles and policies, see the Security Pillar of the AWS Well-Architected Framework.

Resources

Related documents:

- AWS managed policies for job functions
- AWS multiple account billing strategy
- AWS Well-Architected Framework Security Pillar
- AWS Identity and Access Management (IAM)
- AWS Identity and Access Management policies

Related videos:

- Why use Identity and Access Management

Related examples:

- Control access to AWS Regions using IAM policies
- Starting your Cloud Financial Management journey: Cloud cost operations

COST02-BP05 Implement cost controls

Implement controls based on organization policies and defined groups and roles. These certify that costs are only incurred as defined by organization requirements such as control access to regions or resource types.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

A common first step in implementing cost controls is to set up notifications when cost or usage events occur outside of policies. You can act quickly and verify if corrective action is required without restricting or negatively impacting workloads or new activity. After you know the workload and environment limits, you can enforce governance. AWS Budgets allows you to set notifications and define monthly budgets for your AWS costs, usage, and commitment discounts (Savings Plans and Reserved Instances). You can create budgets at an aggregate cost level (for example, all costs), or at a more granular level where you include only specific dimensions such as linked accounts, services, tags, or Availability Zones.

Once you set up your budget limits with AWS Budgets, use AWS Cost Anomaly Detection to reduce your unexpected cost. AWS Cost Anomaly Detection is a cost management service that uses machine learning to continually monitor your cost and usage to detect unusual spends. It helps you identify anomalous spend and root causes, so you can quickly take action. First, create a cost monitor in AWS Cost Anomaly Detection, then choose your alerting preference by setting up a dollar threshold (such as an alert on anomalies with impact greater than $1,000). Once you receive alerts, you can analyze the root cause behind the anomaly and impact on your costs. You can also monitor and perform your own anomaly analysis in AWS Cost Explorer.

Enforce governance policies in AWS through AWS Identity and Access Management and AWS Organizations Service Control Policies (SCP). IAM allows you to securely manage access to AWS services and resources. Using IAM, you can control who can create or manage AWS resources,

the type of resources that can be created, and where they can be created. This minimizes the possibility of resources being created outside of the defined policy. Use the roles and groups created previously and assign IAM policies to enforce the correct usage. SCP offers central control over the maximum available permissions for all accounts in your organization, keeping your accounts stay within your access control guidelines. SCPs are available only in an organization that has all features turned on, and you can configure the SCPs to either deny or allow actions for member accounts by default. For more details on implementing access management, see the Well-Architected Security Pillar whitepaper.

Governance can also be implemented through management of AWS service quotas. By ensuring service quotas are set with minimum overhead and accurately maintained, you can minimize resource creation outside of your organization's requirements. To achieve this, you must understand how quickly your requirements can change, understand projects in progress (both creation and decommission of resources), and factor in how fast quota changes can be implemented. Service quotas can be used to increase your quotas when required.

Implementation steps

- **Implement notifications on spend:** Using your defined organization policies, create AWS Budgets to notify you when spending is outside of your policies. Configure multiple cost budgets, one for each account, which notify you about overall account spending. Configure additional cost budgets within each account for smaller units within the account. These units vary depending on your account structure. Some common examples are AWS Regions, workloads (using tags), or AWS services. Configure an email distribution list as the recipient for notifications, and not an individual's email account. You can configure an actual budget for when an amount is exceeded, or use a forecasted budget for notifying on forecasted usage. You can also preconfigure AWS Budget Actions that can enforce specific IAM or SCP policies, or stop target Amazon EC2 or Amazon RDS instances. Budget Actions can be started automatically or require workflow approval.
- **Implement notifications on anomalous spend:** Use AWS Cost Anomaly Detection to reduce your surprise costs in your organization and analyze root cause of potential anomalous spend. Once you create cost monitor to identify unusual spend at your specified granularity and configure notifications in AWS Cost Anomaly Detection, it sends you alert when unusual spend is detected. This will allow you to analyze root cause behind the anomaly and understand the impact on your cost. Use AWS Cost Categories while configuring AWS Cost Anomaly Detection to identify which project team or business unit team can analyze the root cause of the unexpected cost and take timely necessary actions.

- **Implement controls on usage:** Using your defined organization policies, implement IAM policies and roles to specify which actions users can perform and which actions they cannot. Multiple organizational policies may be included in an AWS policy. In the same way that you defined policies, start broadly and then apply more granular controls at each step. Service limits are also an effective control on usage. Implement the correct service limits on all your accounts.

Resources

Related documents:

- AWS managed policies for job functions
- AWS multiple account billing strategy
- Control access to AWS Regions using IAM policies
- AWS Budgets
- AWS Cost Anomaly Detection
- Control Your AWS Costs

Related videos:

- How can I use AWS Budgets to track my spending and usage

Related examples:

- Example IAM access management policies
- Example service control policies
- AWS Budgets Actions
- Create IAM Policy to control access to Amazon EC2 resources using Tags
- Restrict the access of IAM Identity to specific Amazon EC2 resources
- Slack integrations for Cost Anomaly Detection using Amazon Q Developer in chat applications

COST02-BP06 Track project lifecycle

Track, measure, and audit the lifecycle of projects, teams, and environments to avoid using and paying for unnecessary resources.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

By effectively tracking the project lifecycle, organizations can achieve better cost control through enhanced planning, management, and resource optimization. The insights gained through tracking are invaluable for making informed decisions that contribute to the cost-effectiveness and overall success of the project.

Tracking the entire lifecycle of the workload helps you understand when workloads or workload components are no longer required. The existing workloads and components may appear to be in use, but when AWS releases new services or features, they can be decommissioned or adopted. Check the previous stages of workloads. After a workload is in production, previous environments can be decommissioned or greatly reduced in capacity until they are required again.

You can tag resources with a timeframe or reminder to pin the time that the workload was reviewed. For example, if the development environment was last reviewed months ago, it could be a good time to review it again to explore if new services can be adopted or if the environment is in use. You can group and tag your applications with myApplications on AWS to manage and track metadata such as criticality, environment, last reviewed, and cost center. You can both track your workload's lifecycle and monitor and manage the cost, health, security posture, and performance of your applications.

AWS provides various management and governance services you can use for entity lifecycle tracking. You can use AWS Config or AWS Systems Manager to provide a detailed inventory of your AWS resources and configuration. It is recommended that you integrate with your existing project or asset management systems to keep track of active projects and products within your organization. Combining your current system with the rich set of events and metrics provided by AWS allows you to build a view of significant lifecycle events and proactively manage resources to reduce unnecessary costs.

Similar to Application Lifecycle Management (ALM), tracking project lifecycle should involve multiple processes, tools, and teams working together, such as design and development, testing, production, support, and workload redundancy.

By carefully monitoring each phase of a project's lifecycle, organizations gain crucial insights and enhanced control, facilitating successful project planning, implementation, and completion. This careful oversight verifies that projects not only meet quality standards, but are delivered on time and within budget, promoting overall cost efficiency.

For more details on implementing entity lifecycle tracking, see AWS Well-Architected Operational Excellence Pillar whitepaper.

Implementation steps

- **Establish project lifecycle monitoring process:** The Cloud Center of Excellence team must establish project lifecycle monitoring process. Establish a structured and systematic approach to monitoring workloads in order to improve control, visibility, and performance of the projects. Make the monitoring process transparent, collaborative, and focused on continuous improvement to maximize its effectiveness and value.
- **Perform workload reviews:** As defined by your organizational policies, set up a regular cadence to audit your existing projects and perform workload reviews. The amount of effort spent in the audit should be proportional to the approximate risk, value, or cost to the organization. Key areas to include in the audit would be risk to the organization of an incident or outage, value, or contribution to the organization (measured in revenue or brand reputation), cost of the workload (measured as total cost of resources and operational costs), and usage of the workload (measured in number of organization outcomes per unit of time). If these areas change over the lifecycle, adjustments to the workload are required, such as full or partial decommissioning.

Resources

Related documents:

- Guidance for Tagging on AWS
- What Is ALM (Application Lifecycle Management)?
- AWS managed policies for job functions

Related examples:

- Control access to AWS Regions using IAM policies

Related Tools

- AWS Config
- AWS Systems Manager
- AWS Budgets
- AWS Organizations

- AWS CloudFormation

COST 3. How do you monitor your cost and usage?

Establish policies and procedures to monitor and appropriately allocate your costs. This permits you to measure and improve the cost efficiency of this workload.

Best practices

- COST03-BP01 Configure detailed information sources
- COST03-BP02 Add organization information to cost and usage
- COST03-BP03 Identify cost attribution categories
- COST03-BP04 Establish organization metrics
- COST03-BP05 Configure billing and cost management tools
- COST03-BP06 Allocate costs based on workload metrics

COST03-BP01 Configure detailed information sources

Set up cost management and reporting tools for enhanced analysis and transparency of cost and usage data. Configure your workload to create log entries that facilitate the tracking and segregation of costs and usage.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Detailed billing information such as hourly granularity in cost management tools allow organizations to track their consumptions with further details and help them to identify some of the cost increase reasons. These data sources provide the most accurate view of cost and usage across your entire organization.

You can use AWS Data Exports to create exports of the AWS Cost and Usage Report (CUR) 2.0. This is the new and recommended way to receive your detailed cost and usage data from AWS. It provides daily or hourly usage granularity, rates, costs, and usage attributes for all chargeable AWS services (the same information as CUR), along with some improvements. All possible dimensions are in the CUR such as tagging, location, resource attributes, and account IDs.

You can create 4 types of exports:

- **Standard data export:** A customized export of a table that delivers to Amazon S3 on a recurring basis.
- **Carbon emissions export:** An export of your carbon emissions data to Amazon S3.
- **Cost and usage dashboard:** An export and integration to Quick to deploy a pre-built cost and usage dashboard.
- **Legacy data export:** An export of the legacy AWS Cost and Usage Report (CUR).

You can create data exports with the following customizations:

- Include resource IDs
- Split cost allocation data
- Hourly granularity
- Versioning
- Compression type and file format

For your workloads that run containers on Amazon ECS or Amazon EKS, enable split cost allocation data so that you can allocate your container costs to individual business units and teams, based on how your container workloads consume shared compute and memory resources. Split cost allocation data introduces cost and usage data for new container-level resources to AWS Cost and Usage Report. Split cost allocation data is calculated by computing the cost of individual ECS services and tasks running on the cluster.

A cost and usage dashboard exports the cost and usage dashboard table to an S3 bucket on a recurring basis and deploys a prebuilt cost and usage dashboard to Quick. Use this option if you want to quickly deploy a dashboard of your cost and usage data without the ability for customization.

If desired, you can still export CUR in legacy mode, where you can integrate other processing services such as AWS Glue to prepare the data for analysis and perform data analysis with Amazon Athena using SQL to query the data.

Implementation steps

- **Create data exports:** Create customized exports with the data you want and control the schema of your exports. Create billing and cost management data exports using basic SQL, and visualize

your billing and cost management data by integrating with Quick. You can also export your data in standard mode to analyze your data with other processing tools like Amazon Athena.

- **Configure the cost and usage report:** Using the billing console, configure at least one cost and usage report. Configure a report with hourly granularity that includes all identifiers and resource IDs. You can also create other reports with different granularities to provide higher-level summary information.
- **Configure hourly granularity in Cost Explorer:** To access cost and usage data with hourly granularity for the past 14 days, consider enabling hourly and resource level data in the billing console.
- **Configure application logging:** Verify that your application logs each business outcome that it delivers so it can be tracked and measured. Ensure that the granularity of this data is at least hourly so it matches with the cost and usage data. For more details on logging and monitoring, see Well-Architected Operational Excellence Pillar.

Resources

Related documents:

- AWS Data Exports
- AWS Glue
- Quick
- AWS Cost Management Pricing
- Tagging AWS resources
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

Related examples:

- AWS Account Setup
- Data Exports for AWS Billing and Cost Management
- AWS Cost Explorer Common Use Cases

COST03-BP02 Add organization information to cost and usage

Define a tagging schema based on your organization, workload attributes, and cost allocation categories so that you can filter and search for resources or monitor cost and usage in cost management tools. Implement consistent tagging across all resources where possible by purpose, team, environment, or other criteria relevant to your business.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implement tagging in AWS to add organization information to your resources, which will then be added to your cost and usage information. A tag is a key-value pair — the key is defined and must be unique across your organization, and the value is unique to a group of resources. An example of a key-value pair is the key is `Environment`, with a value of `Production`. All resources in the production environment will have this key-value pair. Tagging allows you categorize and track your costs with meaningful, relevant organization information. You can apply tags that represent organization categories (such as cost centers, application names, projects, or owners), and identify workloads and characteristics of workloads (such as test or production) to attribute your costs and usage throughout your organization.

When you apply tags to your AWS resources (such as Amazon Elastic Compute Cloud instances or Amazon Simple Storage Service buckets) and activate the tags, AWS adds this information to your Cost and Usage Reports. You can run reports and perform analysis on tagged and untagged resources to allow greater compliance with internal cost management policies and ensure accurate attribution.

Creating and implementing an AWS tagging standard across your organization's accounts helps you manage and govern your AWS environments in a consistent and uniform manner. Use Tag Policies in AWS Organizations to define rules for how tags can be used on AWS resources in your accounts in AWS Organizations. Tag Policies allow you to easily adopt a standardized approach for tagging AWS resources

AWS Tag Editor allows you to add, delete, and manage tags of multiple resources. With Tag Editor, you search for the resources that you want to tag, and then manage tags for the resources in your search results.

AWS Cost Categories allows you to assign organization meaning to your costs, without requiring tags on resources. You can map your cost and usage information to unique internal organization structures. You define category rules to map and categorize costs using billing dimensions, such as

accounts and tags. This provides another level of management capability in addition to tagging. You can also map specific accounts and tags to multiple projects.

Implementation steps

- **Define a tagging schema:** Gather all stakeholders from across your business to define a schema. This typically includes people in technical, financial, and management roles. Define a list of tags that all resources must have, as well as a list of tags that resources should have. Verify that the tag names and values are consistent across your organization.
- **Tag resources:** Using your defined cost attribution categories, place tags on all resources in your workloads according to the categories. Use tools such as the CLI, Tag Editor, or AWS Systems Manager to increase efficiency.
- **Implement AWS Cost Categories:** You can create Cost Categories without implementing tagging. Cost categories use the existing cost and usage dimensions. Create category rules from your schema and implement them into cost categories.
- **Automate tagging:** To verify that you maintain high levels of tagging across all resources, automate tagging so that resources are automatically tagged when they are created. Use services such as AWS CloudFormation to verify that resources are tagged when created. You can also create a custom solution to tag automatically using Lambda functions or use a microservice that scans the workload periodically and removes any resources that are not tagged, which is ideal for test and development environments.
- **Monitor and report on tagging:** To verify that you maintain high levels of tagging across your organization, report and monitor the tags across your workloads. You can use AWS Cost Explorer to view the cost of tagged and untagged resources, or use services such as Tag Editor. Regularly review the number of untagged resources and take action to add tags until you reach the desired level of tagging.

Resources

Related documents:

- Tagging Best Practices
- AWS CloudFormation Resource Tag
- AWS Cost Categories
- Tagging AWS resources
- Analyzing your costs with AWS Budgets

- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

Related videos:

- How can I tag my AWS resources to divide up my bill by cost center or project
- Tagging AWS Resources

COST03-BP03 Identify cost attribution categories

Identify organization categories such as business units, departments or projects that could be used to allocate cost within your organization to the internal consuming entities. Use those categories to enforce spend accountability, create cost awareness and drive effective consumption behaviors.

Level of risk exposed if this best practice is not established: High

Implementation guidance

The process of categorizing costs is crucial in budgeting, accounting, financial reporting, decision making, benchmarking, and project management. By classifying and categorizing expenses, teams can gain a better understanding of the types of costs they incur throughout their cloud journey helping teams make informed decisions and manage budgets effectively.

Cloud spend accountability establishes a strong incentive for disciplined demand and cost management. The result is significantly greater cloud cost savings for organizations that allocate most of their cloud spend to consuming business units or teams. Moreover, allocating cloud spend helps organizations adopt more best practices of centralized cloud governance.

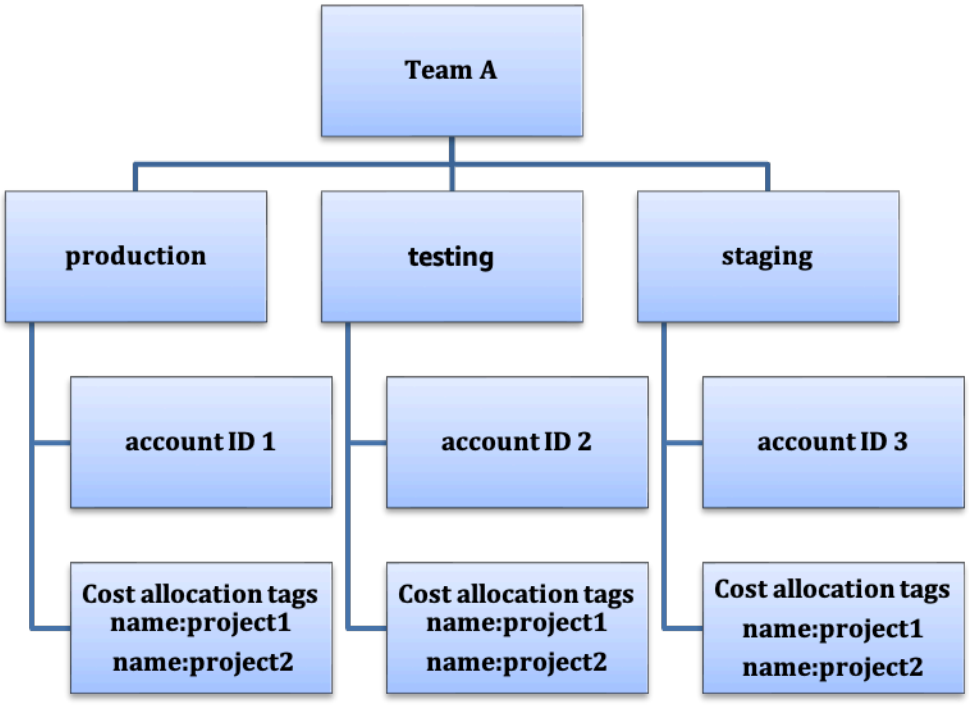
Work with your finance team and other relevant stakeholders to understand the requirements of how costs must be allocated within your organization during your regular cadence calls. Workload costs must be allocated throughout the entire lifecycle, including development, testing, production, and decommissioning. Understand how the costs incurred for learning, staff development, and idea creation are attributed in the organization. This can be helpful to correctly allocate accounts used for this purpose to training and development budgets instead of generic IT cost budgets.

After defining your cost attribution categories with stakeholders in your organization, use AWS Cost Categories to group your cost and usage information into meaningful categories in the AWS

Cloud, such as cost for a specific project, or AWS accounts for departments or business units. You can create custom categories and map your cost and usage information into these categories based on rules you define using various dimensions such as account, tag, service, or charge type. Once cost categories are set up, you can view your cost and usage information by these categories, which allows your organization to make better strategic and purchasing decisions. These categories are visible in AWS Cost Explorer, AWS Budgets, and AWS Cost and Usage Report as well.

For example, create cost categories for your business units (DevOps team), and under each category create multiple rules (rules for each sub category) with multiple dimensions (AWS accounts, cost allocation tags, services or charge type) based on your defined groupings. With cost categories, you can organize your costs using a rule-based engine. The rules that you configure organize your costs into categories. Within these rules, you can filter by using multiple dimensions for each category such as specific AWS accounts, AWS services, or charge types. You can then use these categories across multiple products in the AWS Billing and Cost Management and Cost Management console. This includes AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, and AWS Cost Anomaly Detection.

As an example, the following diagram displays how to group your costs and usage information in your organization by having multiple teams (cost category), multiple environments (rules), and each environment having multiple resources or assets (dimensions).



Cost and usage organization chart

You can create groupings of costs using cost categories as well. After you create the cost categories (allowing up to 24 hours after creating a cost category for your usage records to be updated with values), they appear in AWS Cost Explorer, AWS Budgets, AWS Cost and Usage Report, and AWS Cost Anomaly Detection. In AWS Cost Explorer and AWS Budgets, a cost category appears as an additional billing dimension. You can use this to filter for the specific cost category value, or group by the cost category.

Implementation steps

- **Define your organization categories:** Meet with internal stakeholders and business units to define categories that reflect your organization's structure and requirements. These categories should directly map to the structure of existing financial categories, such as business unit, budget, cost center, or department. Look at the outcomes the cloud delivers for your business such as training or education, as these are also organization categories.
- **Define your functional categories:** Meet with internal stakeholders and business units to define categories that reflect the functions that you have within your business. This may be the workload or application names, and the type of environment, such as production, testing, or development.
- **Define AWS Cost Categories:** Create cost categories to organize your cost and usage information by using AWS Cost Categories and map your AWS cost and usage into meaningful categories. Multiple categories can be assigned to a resource, and a resource can be in multiple different categories, so define as many categories as needed so that you can manage your costs within the categorized structure using AWS Cost Categories.

Resources

Related documents:

- Tagging AWS resources
- Using Cost Allocation Tags
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports
- AWS Cost Categories
- Managing your costs with AWS Cost Categories
- Creating cost categories

- Tagging cost categories
- Splitting charges within cost categories
- AWS Cost Categories Features

Related examples:

- Organize your cost and usage data with AWS Cost Categories
- Managing your costs with AWS Cost Categories

COST03-BP04 Establish organization metrics

Establish the organization metrics that are required for this workload. Example metrics of a workload are customer reports produced, or web pages served to customers.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Understand how your workload's output is measured against business success. Each workload typically has a small set of major outputs that indicate performance. If you have a complex workload with many components, then you can prioritize the list, or define and track metrics for each component. Work with your teams to understand which metrics to use. This unit will be used to understand the efficiency of the workload, or the cost for each business output.

Implementation steps

- **Define workload outcomes:** Meet with the stakeholders in the business and define the outcomes for the workload. These are a primary measure of customer usage and must be business metrics and not technical metrics. There should be a small number of high-level metrics (less than five) per workload. If the workload produces multiple outcomes for different use cases, then group them into a single metric.
- **Define workload component outcomes:** Optionally, if you have a large and complex workload, or can easily break your workload into components (such as microservices) with well-defined inputs and outputs, define metrics for each component. The effort should reflect the value and cost of the component. Start with the largest components and work towards the smaller components.

Resources

Related documents:

- Tagging AWS resources
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

COST03-BP05 Configure billing and cost management tools

Configure cost management tools that meet your organization's policies to manage and optimize cloud spending. This includes services, tools, and resources to organize and track cost and usage data, enhance control through consolidated billing and access permission, improve planning through budgeting and forecasts, receive notifications or alerts, and lower cost with resources and pricing optimizations.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To establish strong accountability, consider your account strategy first as part of your cost allocation strategy. Get this right, and you may not need to go any further. Otherwise, there can be unawareness and further pain points.

To encourage accountability of cloud spend, grant users access to tools that provide visibility into their costs and usage. AWS recommends that you configure all workloads and teams for the following purposes:

- **Organize:** Establish your cost allocation and governance baseline with your own tagging strategy and taxonomy. Create multiple AWS Accounts with tools such as AWS Control Tower or AWS Organization. Tag the supported AWS resources and categorize them meaningfully based on your organization structure (business units, departments, or projects). Tag account names for specific cost centers and map them with AWS Cost Categories to group accounts for business units to their cost centers so that business unit owner can see multiple accounts' consumption in one place.
- **Access:** Track organization-wide billing information in consolidated billing. Verify the right stakeholders and business owners have access.

- **Control:** Build effective governance mechanisms with the right guardrails to prevent unexpected scenarios when using Service Control Policies (SCP), tag policies, IAM policies and budget alerts. For example, you can allow teams to create specific resources in preferred regions only by using effective control mechanisms and prevent resource creations without specific tag (such as cost-center).
- **Current state:** Configure a dashboard that shows current levels of cost and usage. The dashboard should be available in a highly visible place within the work environment like an operations dashboard. You can export data and use the Cost and Usage Dashboard from the AWS Cost Optimization Hub or any supported product to create this visibility. You may need to create different dashboards for different personas. For example, manager dashboard may differ from an engineering dashboard.
- **Notifications:** Provide notifications when cost or usage exceeds defined limits and anomalies occur with AWS Budgets or AWS Cost Anomaly Detection.
- **Reports:** Summarize all cost and usage information. Raise awareness and accountability of your cloud spend with detailed, attributable cost data. Create reports that are relevant to the team consuming them and contain recommendations.
- **Tracking:** Show the current cost and usage against configured goals or targets.
- **Analysis:** Allow team members to perform custom and deep analysis down to the hourly, daily or monthly granularity with different filters (resource, account, tag, etc.).
- **Inspect:** Stay up to date with your resource deployment and cost optimization opportunities. Get notifications using Amazon CloudWatch, Amazon SNS, or Amazon SES for resource deployments at the organization level. Review cost optimization recommendations with AWS Trusted Advisor or AWS Compute Optimizer.
- **Trend reports:** Display the variability in cost and usage over the required period with the required granularity.
- **Forecasts:** Show estimated future costs, estimate your resource usage, and spend with forecast dashboards you create.

You can use AWS Cost Optimization Hub to understand potential cost-saving opportunities consolidated from a centralized location and create data exports for integration with Amazon Athena. You can also use the AWS Cost Optimization Hub to deploy the Cost and Usage Dashboard, which utilizes Quick for interactive cost analysis and secure cost insight sharing.

If you don't have essential skills or bandwidth in your organization, you can work with AWS ProServ, AWS Managed Services (AMS), or AWS Partners. You can also use third-party tools but ensure you validate the value proposition.

Implementation steps

- **Allow team-based access to tools:** Configure your accounts and create groups that have access to the required cost and usage reports for their consumptions and use AWS Identity and Access Management to control access to the tools such as AWS Cost Explorer. These groups must include representatives from all teams that own or manage an application. This certifies that every team has access to their cost and usage information to track their consumption.
- **Organize Costs Tags and Categories:** organize your costs across teams, business units, applications, environments, and projects. Use resource tags to organize costs, by cost allocation tags. Create Cost Categories based on the dimensions by using tags, accounts, services, etc. to map your costs.
- **Configure AWS Budgets:** Configure AWS Budgets on all accounts for your workloads. Set budgets for the overall account spend, and budgets for the workloads by using tags and cost categories. Configure notifications in AWS Budgets to receive alerts for when you exceed your budgeted amounts, or when your estimated costs exceed your budgets.
- **Configure AWS Cost Anomaly Detection:** Use AWS Cost Anomaly Detection for your accounts, core services or cost categories you created to monitor your cost and usage and detect unusual spends. You can receive alerts individually in aggregated reports and receive alerts in an email or an Amazon SNS topic which allows you to analyze and determine the root cause of the anomaly and identify the factor that is driving the cost increase.
- **Use cost analysis tools:** Configure AWS Cost Explorer for your workload and accounts to visualize your cost data for further analysis. Create a dashboard for the workload that tracks overall spend, key usage metrics for the workload, and forecast of future costs based on your historical cost data.
- **Use cost-saving analysis tools:** Use AWS Cost Optimization Hub to identify savings opportunities with tailored recommendations including deleting unused resources, rightsizing, savings Plans, reservations and compute optimizer recommendations.
- **Configure advanced tools:** You can optionally create visuals to facilitate interactive analysis and sharing of cost insights. With Data Exports on AWS Cost Optimization Hub, you can create cost and usage dashboard powered by Quick for your organization that provides additional detail and granularity. You can also implement advanced analysis capability by using data exports in

Amazon Athena for advanced queries, and create dashboards on Quick. Work with AWS Partners to adopt cloud management solutions for consolidated cloud bill monitoring and optimization.

Resources

Related documents:

- What is AWS Billing and Cost Management and Cost Management?
- Establishing your best practice AWS environment
- Best Practices for Tagging AWS Resources
- Tagging your AWS resources
- AWS Cost Categories
- Analyzing your costs with AWS Budgets
- Analyzing your costs with AWS Cost Explorer
- What is AWS Data Exports?

Related videos:

- Deploying Cloud Intelligence Dashboards
- Get Alerts on any FinOps or Cost Optimization Metric or KPI

Related examples:

- Cost and Usage Dashboard powered by Quick
- AWS Cost and Usage Governance Workshop

COST03-BP06 Allocate costs based on workload metrics

Allocate the workload's costs based on usage metrics or business outcomes to measure workload cost efficiency. Implement a process to analyze the cost and usage data with analytics services, which can provide insight and charge back capability.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Cost optimization means delivering business outcomes at the lowest price point, which can only be achieved by allocating workload costs based on workload metrics (measured by workload efficiency). Monitor the defined workload metrics through log files or other application monitoring. Combine this data with the workload's costs, which can be obtained by looking at costs with a specific tag value or account ID. Perform this analysis at the hourly level. Your efficiency typically changes if you have static cost components (for example, a backend database running permanently) with a varying request rate (for example, usage peaks at nine in the morning to five in the evening, with few requests at night). Understanding the relationship between the static and variable costs helps you focus your optimization activities.

Creating workload metrics for shared resources may be challenging compared to resources like containerized applications on Amazon Elastic Container Service (Amazon ECS) and Amazon API Gateway. However, there are certain ways you can categorize usage and track cost. If you need to track Amazon ECS and AWS Batch shared resources, you can enable split cost allocation data in AWS Cost Explorer. With split cost allocation data, you can understand and optimize the cost and usage of your containerized applications and allocate application costs back to individual business entities based on how shared compute and memory resources are consumed.

Implementation steps

- **Allocate costs to workload metrics:** Using the defined metrics and configured tags, create a metric that combines the workload output and workload cost. Use analytics services such as Amazon Athena and Amazon Quick to create an efficiency dashboard for the overall workload and any components.

Resources

Related documents:

- Tagging AWS resources
- Analyzing your costs with AWS Budgets
- Analyzing your costs with Cost Explorer
- Managing AWS Cost and Usage Reports

Related examples:

- Improve cost visibility of Amazon ECS and AWS Batch with AWS Split Cost Allocation Data

COST 4. How do you decommission resources?

Implement change control and resource management from project inception to end-of-life. This ensures you shut down or terminates unused resources to reduce waste.

Best practices

- COST04-BP01 Track resources over their lifetime
- COST04-BP02 Implement a decommissioning process
- COST04-BP03 Decommission resources
- COST04-BP04 Decommission resources automatically
- COST04-BP05 Enforce data retention policies

COST04-BP01 Track resources over their lifetime

Define and implement a method to track resources and their associations with systems over their lifetime. You can use tagging to identify the workload or function of the resource.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Decommission workload resources that are no longer required. A common example is resources used for testing: after testing has been completed, the resources can be removed. Tracking resources with tags (and running reports on those tags) can help you identify assets for decommission, as they will not be in use or the license on them will expire. Using tags is an effective way to track resources, by labeling the resource with its function, or a known date when it can be decommissioned. Reporting can then be run on these tags. Example values for feature tagging are `feature-X testing` to identify the purpose of the resource in terms of the workload lifecycle. Another example is using `LifeSpan` or TTL for the resources, such as to-be-deleted tag key name and value to define the time period or specific time for decommissioning.

Implementation steps

- **Implement a tagging scheme:** Implement a tagging scheme that identifies the workload the resource belongs to, verifying that all resources within the workload are tagged accordingly.

Tagging helps you categorize resources by purpose, team, environment, or other criteria relevant to your business. For more detail on tagging uses cases, strategies, and techniques, see AWS Tagging Best Practices.

- **Implement workload throughput or output monitoring:** Implement workload throughput monitoring or alarming, initiating on either input requests or output completions. Configure it to provide notifications when workload requests or outputs drop to zero, indicating the workload resources are no longer used. Incorporate a time factor if the workload periodically drops to zero under normal conditions. For more detail on unused or underutilized resources, see AWS Trusted Advisor Cost Optimization checks.
- **Group AWS resources:** Create groups for AWS resources. You can use AWS Resource Groups to organize and manage your AWS resources that are in the same AWS Region. You can add tags to most of your resources to help identify and sort your resources within your organization. Use Tag Editor add tags to supported resources in bulk. Consider using AWS Service Catalog to create, manage, and distribute portfolios of approved products to end users and manage the product lifecycle.

Resources

Related documents:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS Trusted Advisor Cost Optimization Checks
- Tagging AWS resources
- Publishing Custom Metrics

Related videos:

- How to optimize costs using AWS Trusted Advisor

Related examples:

- Organize AWS resources
- Optimize cost using AWS Trusted Advisor

COST04-BP02 Implement a decommissioning process

Implement a process to identify and decommission unused resources.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Implement a standardized process across your organization to identify and remove unused resources. The process should define the frequency searches are performed and the processes to remove the resource to verify that all organization requirements are met.

Implementation steps

- **Create and implement a decommissioning process:** Work with the workload developers and owners to build a decommissioning process for the workload and its resources. The process should cover the method to verify if the workload is in use, and also if each of the workload resources are in use. Detail the steps necessary to decommission the resource, removing them from service while ensuring compliance with any regulatory requirements. Any associated resources should be included, such as licenses or attached storage. Notify the workload owners that the decommissioning process has been started.

  Use the following decommission steps to guide you on what should be checked as part of your process:

  - **Identify resources to be decommissioned:** Identify resources that are eligible for decommissioning in your AWS Cloud. Record all necessary information and schedule the decommission. In your timeline, be sure to account for if (and when) unexpected issues arise during the process.
  - **Coordinate and communicate:** Work with workload owners to confirm the resource to be decommissioned
  - **Record metadata and create backups:** Record metadata (such as public IPs, Region, AZ, VPC, Subnet, and Security Groups) and create backups (such as Amazon Elastic Block Store snapshots or taking AMI, keys export, and Certificate export) if it is required for the resources in the production environment or if they are critical resources.
  - **Validate infrastructure-as-code:** Determine whether resources were deployed with CloudFormation, Terraform, AWS Cloud Development Kit (AWS CDK), or any other infrastructure-as-code deployment tool so they can be re-deployed if necessary.

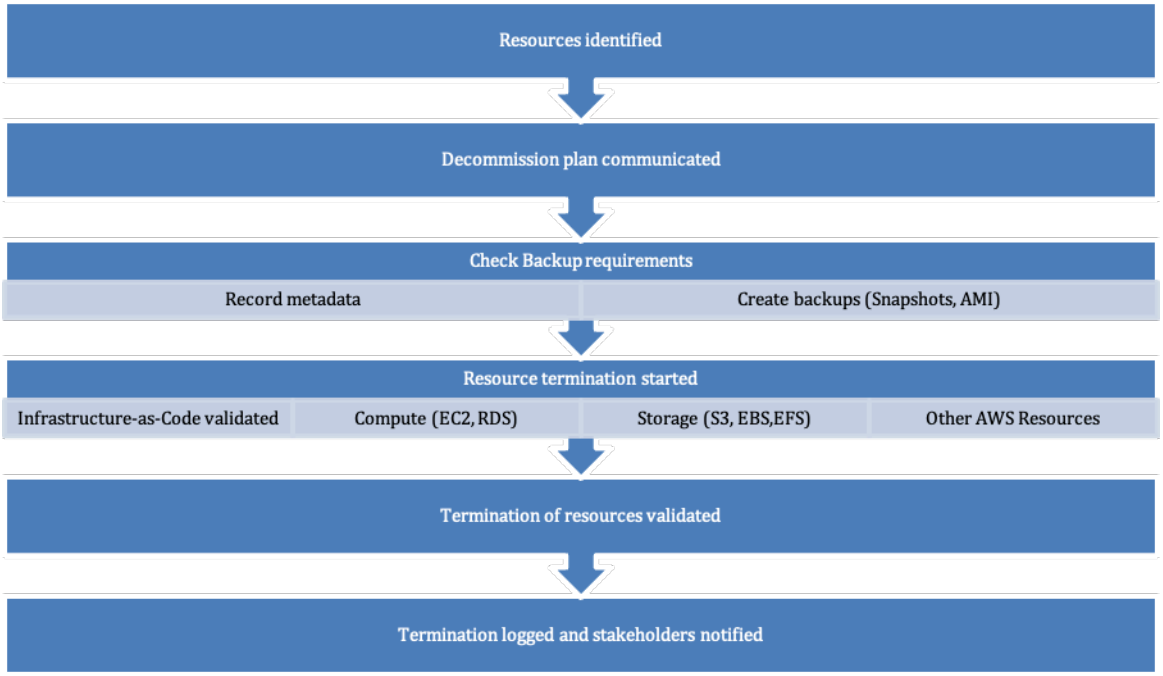
- **Prevent access:** Apply restrictive controls for a period of time, to prevent the use of resources while you determine if the resource is required. Verify that the resource environment can be reverted to its original state if required.
- **Follow your internal decommissioning process:** Follow the administrative tasks and decommissioning process of your organization, like removing the resource from your organization domain, removing the DNS record, and removing the resource from your configuration management tool, monitoring tool, automation tool and security tools.

If the resource is an Amazon EC2 instance, consult the following list. For more detail, see How do I delete or terminate my Amazon EC2 resources?

- Stop or terminate all your Amazon EC2 instances and load balancers. Amazon EC2 instances are visible in the console for a short time after they're terminated. You aren't billed for any instances that aren't in the running state
- Delete your Auto Scaling infrastructure.
- Release all Dedicated Hosts.
- Delete all Amazon EBS volumes and Amazon EBS snapshots.
- Release all Elastic IP addresses.
- Deregister all Amazon Machine Images (AMIs).
- Terminate all AWS Elastic Beanstalk environments.

If the resource is an object in Amazon Glacier storage and if you delete an archive before meeting the minimum storage duration, you will be charged a prorated early deletion fee. Amazon Glacier minimum storage duration depends on the storage class used. For a summary of minimum storage duration for each storage class, see Performance across the Amazon S3 storage classes. For detail on how early deletion fees are calculated, see Amazon S3 pricing.

The following simple decommissioning process flowchart outlines the decommissioning steps. Before decommissioning resources, verify that resources you have identified for decommissioning are not being used by the organization.



Resource decommissioning flow.

Resources

Related documents:

- AWS Auto Scaling
- AWS Trusted Advisor
- AWS CloudTrail

Related videos:

- Delete CloudFormation stack but retain some resources
- Find out which user launched Amazon EC2 instance

Related examples:

- Delete or terminate Amazon EC2 resources
- Find out which user launched an Amazon EC2 instance

COST04-BP03 Decommission resources

Decommission resources initiated by events such as periodic audits, or changes in usage. Decommissioning is typically performed periodically and can be manual or automated.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The frequency and effort to search for unused resources should reflect the potential savings, so an account with a small cost should be analyzed less frequently than an account with larger costs. Searches and decommission events can be initiated by state changes in the workload, such as a product going end of life or being replaced. Searches and decommission events may also be initiated by external events, such as changes in market conditions or product termination.

Implementation steps

- **Decommission resources:** This is the depreciation stage of AWS resources that are no longer needed or ending of a licensing agreement. Complete all final checks completed before moving to the disposal stage and decommissioning resources to prevent any unwanted disruptions like taking snapshots or backups. Using the decommissioning process, decommission each of the resources that have been identified as unused.

Resources

Related documents:

- AWS Auto Scaling
- AWS Trusted Advisor

COST04-BP04 Decommission resources automatically

Design your workload to gracefully handle resource termination as you identify and decommission non-critical resources, resources that are not required, or resources with low utilization.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use automation to reduce or remove the associated costs of the decommissioning process. Designing your workload to perform automated decommissioning will reduce the overall workload

costs during its lifetime. You can use Amazon EC2 Auto Scaling or Application Auto Scaling to perform the decommissioning process. You can also implement custom code using the API or SDK to decommission workload resources automatically.

Modern applications are built serverless-first, a strategy that prioritizes the adoption of serverless services. AWS developed serverless services for all three layers of your stack: compute, integration, and data stores. Using serverless architecture will allow you to save costs during low-traffic periods with scaling up and down automatically.

Implementation steps

- **Implement Amazon EC2 Auto Scaling or Application Auto Scaling:** For resources that are supported, configure them with Amazon EC2 Auto Scaling or Application Auto Scaling. These services can help you optimize your utilization and cost efficiencies when consuming AWS services. When demand drops, these services will automatically remove any excess resource capacity so you avoid overspending.
- **Configure CloudWatch to terminate instances:** Instances can be configured to terminate using CloudWatch alarms. Using the metrics from the decommissioning process, implement an alarm with an Amazon Elastic Compute Cloud action. Verify the operation in a non-production environment before rolling out.
- **Implement code within the workload:** You can use the AWS SDK or AWS CLI to decommission workload resources. Implement code within the application that integrates with AWS and terminates or removes resources that are no longer used.
- **Use serverless services:** Prioritize building serverless architectures and event-driven architecture on AWS to build and run your applications. AWS offers multiple serverless technology services that inherently provide automatically optimized resource utilization and automated decommissioning (scale in and scale out). With serverless applications, resource utilization is automatically optimized and you never pay for over-provisioning.

Resources

Related documents:

- Amazon EC2 Auto Scaling
- Getting Started with Amazon EC2 Auto Scaling
- Application Auto Scaling
- AWS Trusted Advisor

- Serverless on AWS
- Create Alarms to Stop, Terminate, Reboot, or Recover an Instance
- Adding terminate actions to Amazon CloudWatch alarms

Related examples:

- Scheduling automatic deletion of AWS CloudFormation stacks

COST04-BP05 Enforce data retention policies

Define data retention policies on supported resources to handle object deletion per your organizations' requirements. Identify and delete unnecessary or orphaned resources and objects that are no longer required.

Level of risk exposed if this best practice is not established: Medium

Use data retention policies and lifecycle policies to reduce the associated costs of the decommissioning process and storage costs for the identified resources. Defining your data retention policies and lifecycle policies to perform automated storage class migration and deletion will reduce the overall storage costs during its lifetime. You can use Amazon Data Lifecycle Manager to automate the creation and deletion of Amazon Elastic Block Store snapshots and Amazon EBS-backed Amazon Machine Images (AMIs), and use Amazon S3 Intelligent-Tiering or an Amazon S3 lifecycle configuration to manage the lifecycle of your Amazon S3 objects. You can also implement custom code using the API or SDK to create lifecycle policies and policy rules for objects to be deleted automatically.

Implementation steps

- **Use Amazon Data Lifecycle Manager:** Use lifecycle policies on Amazon Data Lifecycle Manager to automate deletion of Amazon EBS snapshots and Amazon EBS-backed AMIs.
- **Set up lifecycle configuration on a bucket:** Use Amazon S3 lifecycle configuration on a bucket to define actions for Amazon S3 to take during an object's lifecycle, as well as deletion at the end of the object's lifecycle, based on your business requirements.

Resources

Related documents:

- AWS Trusted Advisor
- Amazon Data Lifecycle Manager
- How to set lifecycle configuration on Amazon S3 bucket

Related videos:

- Automate Amazon EBS Snapshots with Amazon Data Lifecycle Manager
- Empty an Amazon S3 bucket using a lifecycle configuration rule

Related examples:

- Empty an Amazon S3 bucket using a lifecycle configuration rule

Cost-effective resources

Questions

- COST 5. How do you evaluate cost when you select services?
- COST 6. How do you meet cost targets when you select resource type, size and number?
- COST 7. How do you use pricing models to reduce cost?
- COST 8. How do you plan for data transfer charges?

COST 5. How do you evaluate cost when you select services?

Amazon EC2, Amazon EBS, and Amazon S3 are building-block AWS services. Managed services, such as Amazon RDS and Amazon DynamoDB, are higher level, or application level, AWS services. By selecting the appropriate building blocks and managed services, you can optimize this workload for cost. For example, using managed services, you can reduce or remove much of your administrative and operational overhead, freeing you to work on applications and business-related activities.

Best practices

- COST05-BP01 Identify organization requirements for cost
- COST05-BP02 Analyze all components of the workload
- COST05-BP03 Perform a thorough analysis of each component

- COST05-BP04 Select software with cost-effective licensing
- COST05-BP05 Select components of this workload to optimize cost in line with organization priorities
- COST05-BP06 Perform cost analysis for different usage over time

COST05-BP01 Identify organization requirements for cost

Work with team members to define the balance between cost optimization and other pillars, such as performance and reliability, for this workload.

Level of risk exposed if this best practice is not established: High

Implementation guidance

In most organizations, the information technology (IT) department is comprised of multiple small teams, each with its own agenda and focus area, that reflects the specialisies and skills of its team members. You need to understand your organization's overall objectives, priorities, goals and how each department or project contributes to these objectives. Categorizing all essential resources, including personnel, equipment, technology, materials, and external services, is crucial for achieving organizational objectives and comprehensive budget planning. Adopting this systematic approach to cost identification and understanding is fundamental for establishing a realistic and robust cost plan for the organization.

When selecting services for your workload, it is key that you understand your organization priorities. Create a balance between cost optimization and other AWS Well-Architected Framework pillars, such as performance and reliability. This process should be conducted systematically and regularly to reflect changes in the organization's objectives, market conditions, and operational dynamics. A fully cost-optimized workload is the solution that is most aligned to your organization's requirements, not necessarily the lowest cost. Meet with all teams in your organization, such as product, business, technical, and finance to collect information. Evaluate the impact of tradeoffs between competing interests or alternative approaches to help make informed decisions when determining where to focus efforts or choosing a course of action.

For example, accelerating speed to market for new features may be emphasized over cost optimization, or you may choose a relational database for non-relational data to simplify the effort to migrate a system, rather than migrating to a database optimized for your data type and updating your application.

Implementation steps

- **Identify organization requirements for cost:** Meet with team members from your organization, including those in product management, application owners, development and operational teams, management, and financial roles. Prioritize the Well-Architected pillars for this workload and its components. The output should be a list of the pillars in order. You can also add a weight to each pillar to indicate how much additional focus it has, or how similar the focus is between two pillars.
- **Address the technical debt and document it:** During the workload review, address the technical debt. Document a backlog item to revisit the workload in the future, with the goal of refactoring or re-architecting to optimize it further. It's essential to clearly communicate the trade-offs that were made to other stakeholders.

Resources

Related best practices:

- REL11-BP07 Architect your product to meet availability targets and uptime service level agreements (SLAs)
- OPS01-BP06 Evaluate tradeoffs

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- Cloud products

COST05-BP02 Analyze all components of the workload

Verify every workload component is analyzed, regardless of current size or current costs. The review effort should reflect the potential benefit, such as current and projected costs.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Workload components, which are designed to deliver business value to the organization, may encompass various services. For each component, one might choose specific AWS Cloud services

to address business needs. This selection could be influenced by factors such as familiarity with or prior experience using these services.

After identifying your organization's requirements as mentioned in COST05-BP01 Identify organization requirements for cost, perform a thorough analysis on all components in your workload. Analyze each component considering current and projected costs and sizes. Consider the cost of analysis against any potential workload savings over its lifecycle. The effort expended on the analysis of all components of this workload should correspond to the potential savings or improvements anticipated from optimization of that specific component. For example, if the cost of the proposed resource is $10 per month, and under forecasted loads would not exceed $15 per month, spending a day of effort to reduce costs by 50% (five dollars per month) could exceed the potential benefit over the life of the system. Use a faster and more efficient data-based estimation to create the best overall outcome for this component.

Workloads can change over time, and the right set of services may not be optimal if the workload architecture or usage changes. Analysis for selection of services must incorporate current and future workload states and usage levels. Implementing a service for future workload state or usage may reduce overall costs by reducing or removing the effort required to make future changes. For example, using EMR Serverless might be the appropriate choice initially. However, as consumption for that service increases, transitioning to EMR on EC2 could reduce costs for that component of the workload.

AWS Cost Explorer and the AWS Cost and Usage Reports (CUR) can analyze the cost of a proof of concept (PoC) or running environment. You can also use AWS Pricing Calculator to estimate workload costs.

Write a workflow to be followed by technical teams to review their workloads. Keep this workflow simple, but also cover all the necessary steps to make sure the teams understand each component of the workload and its pricing. Your organization can then follow and customize this workflow based on the specific needs of each team.

1. **List each service in use for your workload:** This is a good starting point. Identify all of the services currently in use and where costs are originate from.
2. **Understand how pricing works for those services:** Understand the pricing model of each service. Different AWS services have different pricing models based on factors like usage volume, data transfer, and feature-specific pricing.
3. **Focus on the services that have unexpected workload costs and that do not align with your expected usage and business outcome:** Identify outliers or services where the cost is not

proportional to the value or usage by using AWS Cost Explorer or AWS Cost and Usage Reports. It's important to correlate costs with business outcomes to prioritize optimization efforts.

4. **AWS Cost Explorer, CloudWatch Logs, VPC Flow Logs, and Amazon S3 Storage Lens to understand the root cause of those high costs:** These tools are instrumental in the diagnosis of high costs. Each service offers a different lens to view and analyze usage and costs. For instance, Cost Explorer helps determine overall cost trends, CloudWatch Logs provides operational insights, VPC Flow Logs displays IP traffic, and Amazon S3 Storage Lens is useful for storage analytics.
5. **Use AWS Budgets to set budgets for certain amounts for services or accounts:** Setting budgets is a proactive way to manage costs. Use AWS Budgets to set custom budget thresholds and receive alerts when costs exceed those thresholds.
6. **Configure Amazon CloudWatch alarms to send billing and usage alerts:** Set up monitoring and alerts for cost and usage metrics. CloudWatch alarms can notify you when certain thresholds are breached, which improves intervention response time.

Facilitate notable enhancement and financial savings over time through strategic review of all workload components and irrespective of their present attributes. The effort invested in this review process should be deliberate, with careful consideration of the potential advantages that might be realized.

Implementation steps

- **List the workload components:** Build a list of your workload's components. Use this list to verify that each component was analyzed. The effort spent should reflect the criticality to the workload as defined by your organization's priorities. Group together resources functionally to improve efficiency (for example, production database storage, if there are multiple databases).
- **Prioritize the component list:** Take the component list and prioritize it in order of effort. This is typically in order of the cost of the component, from most expensive to least expensive or the criticality as defined by your organization's priorities.
- **Perform the analysis:** For each component on the list, review the options and services available, and choose the option that aligns best with your organizational priorities.

Resources

Related documents:

- AWS Pricing Calculator
- AWS Cost Explorer
- Amazon S3 storage classes
- AWS Cloud products

Related videos:

- AWS Cost Optimization Series: CloudWatch

COST05-BP03 Perform a thorough analysis of each component

Look at overall cost to the organization of each component. Calculate the total cost of ownership by factoring in cost of operations and management, especially when using managed services by cloud provider. The review effort should reflect potential benefit (for example, time spent analyzing is proportional to component cost).

Level of risk exposed if this best practice is not established: High

Implementation guidance

Consider the time savings that will allow your team to focus on retiring technical debt, innovation, value-adding features and building what differentiates the business. For example, you might need to lift and shift (also known as rehost) your databases from your on-premises environment to the cloud as rapidly as possible and optimize later. It is worth exploring the possible savings attained by using managed services on AWS that may remove or reduce license costs. Managed services on AWS remove the operational and administrative burden of maintaining a service, such as patching or upgrading the OS, and allow you to focus on innovation and business.

Since managed services operate at cloud scale, they can offer a lower cost per transaction or service. You can make potential optimizations in order to achieve some tangible benefit, without changing the core architecture of the application. For example, you may be looking to reduce the amount of time you spend managing database instances by migrating to a database-as-a-service platform like Amazon Relational Database Service (Amazon RDS) or migrating your application to a fully managed platform like AWS Elastic Beanstalk.

Usually, managed services have attributes that you can set to ensure sufficient capacity. You must set and monitor these attributes so that your excess capacity is kept to a minimum and

performance is maximized. You can modify the attributes of AWS Managed Services using the AWS Management Console or AWS APIs and SDKs to align resource needs with changing demand. For example, you can increase or decrease the number of nodes on an Amazon EMR cluster (or an Amazon Redshift cluster) to scale out or in.

You can also pack multiple instances on an AWS resource to activate higher density usage. For example, you can provision multiple small databases on a single Amazon Relational Database Service (Amazon RDS) database instance. As usage grows, you can migrate one of the databases to a dedicated Amazon RDS database instance using a snapshot and restore process.

When provisioning workloads on managed services, you must understand the requirements of adjusting the service capacity. These requirements are typically time, effort, and any impact to normal workload operation. The provisioned resource must allow time for any changes to occur, provision the required overhead to allow this. The ongoing effort required to modify services can be reduced to virtually zero by using APIs and SDKs that are integrated with system and monitoring tools, such as Amazon CloudWatch.

Amazon RDS, Amazon Redshift, and Amazon ElastiCache provide a managed database service. Amazon Athena, Amazon EMR, and Amazon OpenSearch Service provide a managed analytics service.

AMS is a service that operates AWS infrastructure on behalf of enterprise customers and partners. It provides a secure and compliant environment that you can deploy your workloads onto. AMS uses enterprise cloud operating models with automation to allow you to meet your organization requirements, move into the cloud faster, and reduce your on-going management costs.

Implementation steps

- **Perform a thorough analysis:** Using the component list, work through each component from the highest priority to the lowest priority. For the higher priority and more costly components, perform additional analysis and assess all available options and their long term impact. For lower priority components, assess if changes in usage would change the priority of the component, and then perform an analysis of appropriate effort.
- **Compare managed and unmanaged resources:** Consider the operational cost for the resources you manage and compare them with AWS managed resources. For example, review your databases running on Amazon EC2 instances and compare with Amazon RDS options (an AWS managed service) or Amazon EMR compared to running Apache Spark on Amazon EC2. When moving from a self-managed workload to a AWS fully managed workload, research your options

carefully. The three most important factors to consider are the type of managed service you want to use, the process you will use to migrate your data and understand the AWS shared responsibility model.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- AWS Cloud products
- AWS Shared Responsibility Model

Related videos:

- Why move to a managed database?
- What is Amazon EMR and how can I use it for processing data?

Related examples:

- Why to move to a managed database
- Consolidate data from identical SQL Server databases into a single Amazon RDS for SQL Server database using AWS DMS
- Deliver data at scale to Amazon Managed Streaming for Apache Kafka (Amazon MSK)
- Migrate an ASP.NET web application to AWS Elastic Beanstalk

COST05-BP04 Select software with cost-effective licensing

Open-source software eliminates software licensing costs, which can contribute significant costs to workloads. Where licensed software is required, avoid licenses bound to arbitrary attributes such as CPUs, look for licenses that are bound to output or outcomes. The cost of these licenses scales more closely to the benefit they provide.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Open source originated in the context of software development to indicate that the software complies with certain free distribution criteria. Open source software is composed of source code that anyone can inspect, modify, and enhance. Based on business requirements, skill of engineers, forecasted usage, or other technology dependencies, organizations can consider using open source software on AWS to minimize their license costs. In other words, the cost of software licenses can be reduced through the use of open source software. This can have significant impact on workload costs as the size of the workload scales.

Measure the benefits of licensed software against the total cost to optimize your workload. Model any changes in licensing and how they would impact your workload costs. If a vendor changes the cost of your database license, investigate how that impacts the overall efficiency of your workload. Consider historical pricing announcements from your vendors for trends of licensing changes across their products. Licensing costs may also scale independently of throughput or usage, such as licenses that scale by hardware (CPU bound licenses). These licenses should be avoided because costs can rapidly increase without corresponding outcomes.

For instance, operating an Amazon EC2 instance in us-east-1 with a Linux operating system allows you to cut costs by approximately 45%, compared to running another Amazon EC2 instance that runs on Windows.

The AWS Pricing Calculator offers a comprehensive way to compare the costs of various resources with different license options, such as Amazon RDS instances and different database engines. Additionally, the AWS Cost Explorer provides an invaluable perspective for the costs of existing workloads, especially those that come with different licenses. For license management, AWS License Manager offers a streamlined method to oversee and handle software licenses. Customers can deploy and operationalize their preferred open source software in the AWS Cloud.

Implementation steps

- **Analyze license options:** Review the licensing terms of available software. Look for open source versions that have the required functionality, and whether the benefits of licensed software outweigh the cost. Favorable terms align the cost of the software to the benefits it provides.
- **Analyze the software provider:** Review any historical pricing or licensing changes from the vendor. Look for any changes that do not align to outcomes, such as punitive terms for running on specific vendors hardware or platforms. Additionally, look for how they perform audits, and penalties that could be imposed.

Resources

Related documents:

- Open Source at AWS
- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- Cloud products

Related examples:

- Open Source Blogs
- AWS Open Source Blogs
- Optimization and Licensing Assessment

COST05-BP05 Select components of this workload to optimize cost in line with organization priorities

Factor in cost when selecting all components for your workload. This includes using application-level and managed services or serverless, containers, or event-driven architecture to reduce overall cost. Minimize license costs by using open-source software, software that does not have license fees, or alternatives to reduce spending.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Consider the cost of services and options when selecting all components. This includes using application level and managed services, such as Amazon Relational Database Service (Amazon RDS), Amazon DynamoDB, Amazon Simple Notification Service (Amazon SNS), and Amazon Simple Email Service (Amazon SES) to reduce overall organization cost.

Use serverless and containers for compute, such as AWS Lambda and Amazon Simple Storage Service (Amazon S3) for static websites. Containerize your application if possible and use AWS Managed Container Services such as Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS).

Minimize license costs by using open-source software, or software that does not have license fees (for example, Amazon Linux for compute workloads or migrate databases to Amazon Aurora).

You can use serverless or application-level services such as Lambda, Amazon Simple Queue Service (Amazon SQS), Amazon SNS, and Amazon SES. These services remove the need for you to manage a resource and provide the function of code execution, queuing services, and message delivery. The other benefit is that they scale in performance and cost in line with usage, allowing efficient cost allocation and attribution.

Using event-driven architecture is also possible with serverless services. Event-driven architectures are push-based, so everything happens on demand as the event presents itself in the router. This way, you're not paying for continuous polling to check for an event. This means less network bandwidth consumption, less CPU utilization, less idle fleet capacity, and fewer SSL/TLS handshakes.

For more information on serverless, see Well-Architected Serverless Application lens whitepaper.

Implementation steps

- **Select each service to optimize cost:** Using your prioritized list and analysis, select each option that provides the best match with your organizational priorities. Instead of increasing the capacity to meet the demand, consider other options which may give you better performance with lower cost. For example, if you need to review expected traffic for your databases on AWS, consider either increasing the instance size or using Amazon ElastiCache services (Redis or Memcached) to provide cached mechanisms for your databases.
- **Evaluate event-driven architecture:** Using serverless architecture also allows you to build event-driven architecture for distributed microservice-based applications, which helps you build scalable, resilient, agile and cost-effective solutions.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- AWS Serverless
- What is Event-Driven Architecture
- Amazon S3 storage classes
- Cloud products
- Amazon ElastiCache (Redis OSS)

Related examples:

- Getting started with event-driven architecture
- Event-driven architecture
- How Statsig runs 100x more cost-effectively using Amazon ElastiCache (Redis OSS)
- Best practices for working with AWS Lambda functions

COST05-BP06 Perform cost analysis for different usage over time

Workloads can change over time. Some services or features are more cost effective at different usage levels. By performing the analysis on each component over time and at projected usage, the workload remains cost-effective over its lifetime.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

As AWS releases new services and features, the optimal services for your workload may change. Effort required should reflect potential benefits. Workload review frequency depends on your organization requirements. If it is a workload of significant cost, implementing new services sooner will maximize cost savings, so more frequent review can be advantageous. Another initiation for review is change in usage patterns. Significant changes in usage can indicate that alternate services would be more optimal.

If you need to move data into AWS Cloud, you can select any wide variety of services AWS offers and partner tools to help you migrate your data sets, whether they are files, databases, machine images, block volumes, or even tape backups. For example, to move a large amount of data to and from AWS or process data at the edge, you can use one of the AWS purpose-built devices to cost effectively move petabytes of data offline. Another example is for higher data transfer rates, a direct connect service may be cheaper than a VPN which provides the required consistent connectivity for your business.

Based on the cost analysis for different usage over time, review your scaling activity. Analyze the result to see if the scaling policy can be tuned to add instances with multiple instance types and purchase options. Review your settings to see if the minimum can be reduced to serve user requests but with a smaller fleet size, and add more resources to meet the expected high demand.

Perform cost analysis for different usage over time by discussing with stakeholders in your organization and use AWS Cost Explorer's forecast feature to predict the potential impact of

service changes. Monitor usage level launches using AWS Budgets, CloudWatch billing alarms and AWS Cost Anomaly Detection to identify and implement the most cost-effective services sooner.

Implementation steps

- **Define predicted usage patterns:** Working with your organization, such as marketing and product owners, document what the expected and predicted usage patterns will be for the workload. Discuss with business stakeholders about both historical and forecasted cost and usage increases and make sure increases align with business requirements. Identify calendar days, weeks, or months where you expect more users to use your AWS resources, which indicate that you should increase the capacity of the existing resources or adopt additional services to reduce the cost and increase performance.
- **Perform cost analysis at predicted usage:** Using the usage patterns defined, perform analysis at each of these points. The analysis effort should reflect the potential outcome. For example, if the change in usage is large, a thorough analysis should be performed to verify any costs and changes. In other words, when cost increases, usage should increase for business as well.

Resources

Related documents:

- AWS Total Cost of Ownership (TCO) Calculator
- Amazon S3 storage classes
- Cloud products
- Amazon EC2 Auto Scaling
- Cloud Data Migration
- AWS Snow Family

Related videos:

- AWS OpsHub for Snow Family

COST 6. How do you meet cost targets when you select resource type, size and number?

Verify that you choose the appropriate resource size and number of resources for the task at hand. You minimize waste by selecting the most cost effective type, size, and number.

Best practices

- COST06-BP01 Perform cost modeling
- COST06-BP02 Select resource type, size, and number based on data
- COST06-BP03 Select resource type, size, and number automatically based on metrics
- COST06-BP04 Consider using shared resources

COST06-BP01 Perform cost modeling

Identify organization requirements (such as business needs and existing commitments) and perform cost modeling (overall costs) of the workload and each of its components. Perform benchmark activities for the workload under different predicted loads and compare the costs. The modeling effort should reflect the potential benefit. For example, time spent is proportional to component cost.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Perform cost modeling for your workload and each of its components to understand the balance between resources, and find the correct size for each resource in the workload, given a specific level of performance. Understanding cost considerations can inform your organizational business case and decision-making process when evaluating the value realization outcomes for planned workload deployment.

Perform benchmark activities for the workload under different predicted loads and compare the costs. The modeling effort should reflect potential benefit; for example, time spent is proportional to component cost or predicted saving. For best practices, refer to the Review section of the Performance Efficiency Pillar of the AWS Well-Architected Framework.

As an example, to create cost modeling for a workload consisting of compute resources, AWS Compute Optimizer can assist with cost modeling for running workloads. It provides right-sizing

recommendations for compute resources based on historical usage. Make sure CloudWatch Agents are deployed to the Amazon EC2 instances to collect memory metrics which help you with more accurate recommendations within AWS Compute Optimizer. This is the ideal data source for compute resources because it is a free service that uses machine learning to make multiple recommendations depending on levels of risk.

There are multiple services you can use with custom logs as data sources for rightsizing operations for other services and workload components, such as AWS Trusted Advisor, Amazon CloudWatch and Amazon CloudWatch Logs. AWS Trusted Advisor checks resources and flags resources with low utilization which can help you rightsize your resources and create cost modeling.

The following are recommendations for cost modeling data and metrics:

- The monitoring must accurately reflect the user experience. Select the correct granularity for the time period and thoughtfully choose the maximum or 99th percentile instead of the average.
- Select the correct granularity for the time period of analysis that is required to cover any workload cycles. For example, if a two-week analysis is performed, you might be overlooking a monthly cycle of high utilization, which could lead to under-provisioning.
- Choose the right AWS services for your planned workload by considering your existing commitments, selected pricing models for other workloads, and ability to innovate faster and focus on your core business value.

Implementation steps

- **Perform cost modeling for resources:** Deploy the workload or a proof of concept into a separate account with the specific resource types and sizes to test. Run the workload with the test data and record the output results, along with the cost data for the time the test was run. Afterwards, redeploy the workload or change the resource types and sizes and run the test again. Include license fees of any products you may use with these resources and estimated operations (labor or engineer) costs for deploying and managing these resources while creating cost modeling. Consider cost modeling for a period (hourly, daily, monthly, yearly or three years).

Resources

Related documents:

- AWS Auto Scaling
- Identifying Opportunities to Right Size

- Amazon CloudWatch features
- Cost Optimization: Amazon EC2 Right Sizing
- AWS Compute Optimizer
- AWS Pricing Calculator

Related examples:

- Perform a Data-Driven Cost modeling
- Estimate the cost of planned AWS resource configurations
- Choose the right AWS tools

COST06-BP02 Select resource type, size, and number based on data

Select resource size or type based on data about the workload and resource characteristics. For example, compute, memory, throughput, or write intensive. This selection is typically made using a previous (on-premises) version of the workload, using documentation, or using other sources of information about the workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Amazon EC2 provides a wide selection of instance types with different levels of CPU, memory, storage, and networking capacity to fit different use cases. These instance types feature different blends of CPU, memory, storage, and networking capabilities, giving you versatility when selecting the right resource combination for your projects. Every instance type comes in multiple sizes, so that you can adjust your resources based on your workload's demands. To determine which instance type you need, gather details about the system requirements of the application or software that you plan to run on your instance. These details should include the following:

- Operating system
- Number of CPU cores
- GPU cores
- Amount of system memory (RAM)
- Storage type and space
- Network bandwidth requirement

Identify the purpose of compute requirements and which instance is needed, and then explore the various Amazon EC2 instance families. Amazon offers the following instance type families:

- General Purpose
- Compute Optimized
- Memory Optimized
- Storage Optimized
- Accelerated Computing
- HPC Optimized

For a deeper understanding of the specific purposes and use cases that a particular Amazon EC2 instance family can fulfill, see AWS Instance types.

System requirements gathering is critical for you to select the specific instance family and instance type that best serves your needs. Instance type names are comprised of the family name and the instance size. For example, the t2.micro instance is from the T2 family and is micro-sized.

Select resource size or type based on workload and resource characteristics (for example, compute, memory, throughput, or write intensive). This selection is typically made using cost modeling, a previous version of the workload (such as an on-premises version), using documentation, or using other sources of information about the workload (whitepapers or published solutions). Using AWS pricing calculators or cost management tools can assist in making informed decisions about instance types, sizes, and configurations.

Implementation steps

- **Select resources based on data:** Use your cost modeling data to select the anticipated workload usage level, and choose the specified resource type and size. Relying on the cost modeling data, determine the number of virtual CPUs, total memory (GiB), the local instance store volume (GB), Amazon EBS volumes, and the network performance level, taking into account the data transfer rate required for the instance. Always make selections based on detailed analysis and accurate data to optimize performance while managing costs effectively.

Resources

Related documents:

- AWS Instance types

- AWS Auto Scaling
- Amazon CloudWatch features
- Cost Optimization: EC2 Right Sizing

Related videos:

- Selecting the right Amazon EC2 instance for your workloads
- Right size your service

Related examples:

- It just got easier to discover and compare Amazon EC2 instance types

COST06-BP03 Select resource type, size, and number automatically based on metrics

Use metrics from the currently running workload to select the right size and type to optimize for cost. Appropriately provision throughput, sizing, and storage for compute, storage, data, and networking services. This can be done with a feedback loop such as automatic scaling or by custom code in the workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Create a feedback loop within the workload that uses active metrics from the running workload to make changes to that workload. You can use a managed service, such as AWS Auto Scaling, which you configure to perform the right sizing operations for you. AWS also provides APIs, SDKs, and features that allow resources to be modified with minimal effort. You can program a workload to stop-and-start an Amazon EC2 instance to allow a change of instance size or instance type. This provides the benefits of right-sizing while removing almost all the operational cost required to make the change.

Some AWS services have built in automatic type or size selection, such as Amazon Simple Storage Service Intelligent-Tiering. Amazon S3 Intelligent-Tiering automatically moves your data between two access tiers, frequent access and infrequent access, based on your usage patterns.

Implementation steps

- **Increase your observability by configuring workload metrics:** Capture key metrics for the workload. These metrics provide an indication of the customer experience, such as workload output, and align to the differences between resource types and sizes, such as CPU and memory usage. For compute resource, analyze performance data to right size your Amazon EC2 instances. Identify idle instances and ones that are underutilized. Key metrics to look for are CPU usage and memory utilization (for example, 40% CPU utilization at 90% of the time as explained in Rightsizing with AWS Compute Optimizer and Memory Utilization Enabled). Identify instances with a maximum CPU usage and memory utilization of less than 40% over a four-week period. These are the instances to right size to reduce costs. For storage resources such as Amazon S3, you can use Amazon S3 Storage Lens, which allows you to see 28 metrics across various categories at the bucket level, and 14 days of historical data in the dashboard by default. You can filter your Amazon S3 Storage Lens dashboard by summary and cost optimization or events to analyze specific metrics.
- **View rightsizing recommendations:** Use the rightsizing recommendations in AWS Compute Optimizer and the Amazon EC2 rightsizing tool in the Cost Management console, or review AWS Trusted Advisor right-sizing your resources to make adjustments on your workload. It is important to use the right tools when right-sizing different resources and follow right-sizing guidelines whether it is an Amazon EC2 instance, AWS storage classes, or Amazon RDS instance types. For storage resources, you can use Amazon S3 Storage Lens, which gives you visibility into object storage usage, activity trends, and makes actionable recommendations to optimize costs and apply data protection best practices. Using the contextual recommendations that Amazon S3 Storage Lens derives from analysis of metrics across your organization, you can take immediate steps to optimize your storage.
- **Select resource type and size automatically based on metrics:** Using the workload metrics, manually or automatically select your workload resources. For compute resources, configuring AWS Auto Scaling or implementing code within your application can reduce the effort required if frequent changes are needed, and it can potentially implement changes sooner than a manual process. You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you optimize your cost savings for Amazon EC2 instances and determine the desired scale and performance for your application. You can also use an attribute-based instance type selection (ABS) strategy in Auto Scaling Groups (ASG), which lets you express your instance requirements as a set of attributes, such as vCPU, memory, and storage. You can automatically use newer generation instance types when they are released and access a broader range of capacity with Amazon EC2

Spot Instances. Amazon EC2 Fleet and Amazon EC2 Auto Scaling select and launch instances that fit the specified attributes, removing the need to manually pick instance types. For storage resources, you can use the Amazon S3 Intelligent Tiering and Amazon EFS Infrequent Access features, which allow you to select storage classes automatically that deliver automatic storage cost savings when data access patterns change, without performance impact or operational overhead.

Resources

Related documents:

- AWS Auto Scaling
- AWS Right-Sizing
- AWS Compute Optimizer
- Amazon CloudWatch features
- CloudWatch Getting Set Up
- CloudWatch Publishing Custom Metrics
- Getting Started with Amazon EC2 Auto Scaling
- Amazon S3 Storage Lens
- Amazon S3 Intelligent-Tiering
- Amazon EFS Infrequent Access
- Launch an Amazon EC2 Instance Using the SDK

Related videos:

- Right Size Your Services

Related examples:

- Attribute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive scaling with Amazon EC2 Auto Scaling
- Optimize Costs and Gain Visibility into Usage with Amazon S3 Storage Lens

COST06-BP04 Consider using shared resources

For already-deployed services at the organization level for multiple business units, consider using shared resources to increase utilization and reduce total cost of ownership (TCO). Using shared resources can be a cost-effective option to centralize the management and costs by using existing solutions, sharing components, or both. Manage common functions like monitoring, backups, and connectivity either within an account boundary or in a dedicated account. You can also reduce cost by implementing standardization, reducing duplication, and reducing complexity.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Where multiple workloads cause the same function, use existing solutions and shared components to improve management and optimize costs. Consider using existing resources (especially shared ones), such as non-production database servers or directory services, to mitigate cloud costs by following security best practices and organizational regulations. For optimal value realization and efficiency, it is crucial to allocate costs back (using showback and chargeback) to the pertinent areas of the business driving consumption.

Showback refers to reports that break down cloud costs into attributable categories, such as consumers, business units, general ledger accounts, or other responsible entities. The goal of showback is to show teams, business units, or individuals the cost of their consumed cloud resources.

Chargeback means to allocate central service spend to cost units based on a strategy suitable for a specific financial management process. For customers, chargeback charges the cost incurred from one shared services account to different financial cost categories suitable for a customer reporting process. By establishing chargeback mechanisms, you can report costs incurred by different business units, products, and teams.

Workloads can be categorized as critical and non-critical. Based on this classification, use shared resources with general configurations for less critical workloads. To further optimize costs, reserve dedicated servers solely for critical workloads. Share resources or provision them across several accounts to manage them efficiently. Even with distinct development, testing, and production environments, secure sharing is feasible and does not compromise organizational structure.

To improve your understanding and optimize cost and usage for containerized applications, use split cost allocation data which helps you allocate costs to individual business entities based on how the application consumes shared compute and memory resources. Split cost allocation data

helps you achieve task-level showback and chargeback in container workloads running on Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS).

For distributed architectures, build a shared services VPC, which provides centralized access to shared services required by workloads in each of the VPCs. These shared services can include resources such as directory services or VPC endpoints. To reduce administrative overhead and cost, share resources from a central location instead of building them in each VPC.

When you use shared resources, you can save on operational costs, maximize resource utilization, and improve consistency. In a multi-account design, you can host some AWS services centrally and access them using several applications and accounts in a hub to save cost. You can use AWS Resource Access Manager (AWS RAM) to share other common resources, such as VPC subnets and AWS Transit Gateway attachments, AWS Network Firewall, or Amazon SageMaker AI pipelines. In a multi-account environment, use AWS RAM to create a resource once and share it with other accounts.

Organizations should tag shared costs effectively and verify that they do not have a significant portion of their costs untagged or unallocated. If you do not allocate shared costs effectively and no one takes accountability for shared costs management, shared cloud costs can spiral. You should know where you have incurred costs at the resource, workload, team, or organization level, as this knowledge enhances your understanding of the value delivered at the applicable level when compared to the business outcomes achieved. Ultimately, organizations benefit from cost savings as a result of sharing cloud infrastructure. Encourage cost allocation on shared cloud resources to optimize cloud spend.

Implementation steps

- **Evaluate existing resources:** Review existing workloads that use similar services for your workload. Depending on the workload's components, consider existing platforms if business logic or technical requirement allow.
- **Use resource sharing in AWS RAM and restrict accordingly:** Use AWS RAM to share resources with other AWS accounts within your organization. When you share resources, you don't need to duplicate resources in multiple accounts, which minimizes the operational burden of resource maintenance. This process also helps you securely share the resources that you have created with roles and users in your account, as well as with other AWS accounts.
- **Tag resources:** Tag resources that are candidates for cost reporting and categorize them within cost categories. Activate these cost related resource tags for cost allocation to provide visibility of AWS resources usage. Focus on creating an appropriate level of granularity with respect to

cost and usage visibility, and influence cloud consumption behaviors through cost allocation reporting and KPI tracking.

Resources

Related best practices:

- SEC03-BP08 Share resources securely within your organization

Related documents:

- What is AWS Resource Access Manager?
- AWS services that you can use with AWS Organizations
- Shareable AWS resources
- AWS Cost and Usage (CUR) Queries

Related videos:

- AWS Resource Access Manager - granular access control with managed permissions
- How to design your AWS cost allocation strategy
- AWS Cost Categories

Related examples:

- How-to chargeback shared services: An AWS Transit Gateway example
- How to build a chargeback/showback model for Savings Plans using the CUR
- Using VPC Sharing for a Cost-Effective Multi-Account Microservice Architecture
- Improve cost visibility of Amazon EKS with AWS Split Cost Allocation Data
- Improve cost visibility of Amazon ECS and AWS Batch with AWS Split Cost Allocation Data

COST 7. How do you use pricing models to reduce cost?

Use the pricing model that is most appropriate for your resources to minimize expense.

Best practices

- COST07-BP01 Perform pricing model analysis
- COST07-BP02 Choose Regions based on cost
- COST07-BP03 Select third-party agreements with cost-efficient terms
- COST07-BP04 Implement pricing models for all components of this workload
- COST07-BP05 Perform pricing model analysis at the management account level

COST07-BP01 Perform pricing model analysis

Analyze each component of the workload. Determine if the component and resources will be running for extended periods (for commitment discounts) or dynamic and short-running (for spot or on-demand). Perform an analysis on the workload using the recommendations in cost management tools and apply business rules to those recommendations to achieve high returns.

Level of risk exposed if this best practice is not established: High

Implementation guidance

AWS has multiple pricing models that allow you to pay for your resources in the most cost-effective way that suits your organization's needs and depending on product. Work with your teams to determine the most appropriate pricing model. Often your pricing model consists of a combination of multiple options, as determined by your availability

On-Demand Instances allow you to pay for compute or database capacity by the hour or by the second (60 seconds minimum) depending on which instances you run, without long-term commitments or upfront payments.

Savings Plans are a flexible pricing model that offers low prices on Amazon EC2, Lambda, and AWS Fargate usage, in exchange for a commitment to a consistent amount of usage (measured in dollars per hour) over one year or three years terms.

Spot Instances are an Amazon EC2 pricing mechanism that allows you request spare compute capacity at discounted hourly rate (up to 90% off the on-demand price) without upfront commitment.

Reserved Instances allow you up to 75 percent discount by prepaying for capacity. For more details, see Optimizing costs with reservations.

You might choose to include a Savings Plans for the resources associated with the production, quality, and development environments. Alternatively, because sandbox resources are only powered on when needed, you might choose an on-demand model for the resources in that

environment. Use Amazon Spot Instances to reduce Amazon EC2 costs or use Compute Savings Plans to reduce Amazon EC2, Fargate, and Lambda cost. The AWS Cost Explorer recommendations tool provides opportunities for commitment discounts with Saving plans.

If you have been purchasing Reserved Instances for Amazon EC2 in the past or have established cost allocation practices inside your organization, you can continue using Amazon EC2 Reserved Instances for the time being. However, we recommend working on a strategy to use Savings Plans in the future as a more flexible cost savings mechanism. You can refresh Savings Plans (SP) Recommendations in AWS Cost Management to generate new Savings Plans Recommendations at any time. Use Reserved Instances (RI) to reduce Amazon RDS, Amazon Redshift, Amazon ElastiCache, and Amazon OpenSearch Service costs. Saving Plans and Reserved Instances are available in three options: all upfront, partial upfront and no upfront payments. Use the recommendations provided in AWS Cost Explorer RI and SP purchase recommendations.

To find opportunities for Spot workloads, use an hourly view of your overall usage, and look for regular periods of changing usage or elasticity. You can use Spot Instances for various fault-tolerant and flexible applications. Examples include stateless web servers, API endpoints, big data and analytics applications, containerized workloads, CI/CD, and other flexible workloads.

Analyze your Amazon EC2 and Amazon RDS instances whether they can be turned off when you don’t use (after hours and weekends). This approach will allow you to reduce costs by 70% or more compared to using them 24/7. If you have Amazon Redshift clusters that only need to be available at specific times, you can pause the cluster and later resume it. When the Amazon Redshift cluster or Amazon EC2 and Amazon RDS Instance is stopped, the compute billing halts and only the storage charge applies.

Note that On-Demand Capacity reservations (ODCR) are not a pricing discount. Capacity Reservations are charged at the equivalent On-Demand rate, whether you run instances in reserved capacity or not. They should be considered when you need to provide enough capacity for the resources you plan to run. ODCRs don't have to be tied to long-term commitments, as they can be cancelled when you no longer need them, but they can also benefit from the discounts that Savings Plans or Reserved Instances provide.

Implementation steps

- **Analyze workload elasticity:** Using the hourly granularity in Cost Explorer or a custom dashboard, analyze your workload's elasticity. Look for regular changes in the number of instances that are running. Short duration instances are candidates for Spot Instances or Spot Fleet.

- Well-Architected Lab: Cost Explorer
- Well-Architected Lab: Cost Visualization

- **Review existing pricing contracts:** Review current contracts or commitments for long term needs. Analyze what you currently have and how much those commitments are in use. Leverage pre-existing contractual discounts or enterprise agreements. Enterprise Agreements give customers the option to tailor agreements that best suit their needs. For long term commitments, consider reserved pricing discounts, Reserved Instances or Savings Plans for the specific instance type, instance family, AWS Region, and Availability Zones.
- **Perform a commitment discount analysis:** Using Cost Explorer in your account, review the Savings Plans and Reserved Instance recommendations. To verify that you implement the correct recommendations with the required discounts and risk, follow the Well-Architected labs.

Resources

Related documents:

- Accessing Reserved Instance recommendations
- Instance purchasing options
- AWS Enterprise

Related videos:

- Save up to 90% and run production workloads on Spot

Related examples:

- Well-Architected Lab: Cost Explorer
- Well-Architected Lab: Cost Visualization
- Well-Architected Lab: Pricing Models

COST07-BP02 Choose Regions based on cost

Resource pricing may be different in each Region. Identify Regional cost differences and only deploy in Regions with higher costs to meet latency, data residency and data sovereignty requirements. Factoring in Region cost helps you pay the lowest overall price for this workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The AWS Cloud Infrastructure is global, hosted in multiple locations world-wide, and built around AWS Regions, Availability Zones, Local Zones, AWS Outposts, and Wavelength Zones. A Region is a physical location in the world and each Region is a separate geographic area where AWS has multiple Availability Zones. Availability Zones which are multiple isolated locations within each Region consist of one or more discrete data centers, each with redundant power, networking, and connectivity.

Each AWS Region operates within local market conditions, and resource pricing is different in each Region due to differences in the cost of land, fiber, electricity, and taxes, for example. Choose a specific Region to operate a component of or your entire solution so that you can run at the lowest possible price globally. Use AWS Calculator to estimate the costs of your workload in various Regions by searching services by location type (Region, wave length zone and local zone) and Region.

When you architect your solutions, a best practice is to seek to place computing resources closer to users to provide lower latency and strong data sovereignty. Select the geographic location based on your business, data privacy, performance, and security requirements. For applications with global end users, use multiple locations.

Use Regions that provide lower prices for AWS services to deploy your workloads if you have no obligations in data privacy, security and business requirements. For example, if your default Region is Asia Pacific (Sydney) (`ap-southwest-2`), and if there are no restrictions (data privacy, security, for example) to use other Regions, deploying non-critical (development and test) Amazon EC2 instances in US East (N. Virginia) (`us-east-1`) will cost you less.

	Compliance	*Latency*	*Cost*	*Services / Features*
Region 1	✓	15 ms	$$	✓
Region 2	✓	20 ms	$$$	X
Region 3	✓	80 ms	$	✓
Region 4	✓	15 ms	$$	✓
Region 5	✓	20 ms	$$$	X
Region 6	✓	15 ms	$	✓
Region 7	✓	80 ms	$	✓
Region 8	✓	15 ms	$	X

Region feature matrix table

The preceding matrix table shows us that Region 6 is the best option for this given scenario because latency is low compared to other Regions, service is available, and it is the least expensive Region.

Implementation steps

- **Review AWS Region pricing:** Analyze the workload costs in the current Region. Starting with the highest costs by service and usage type, calculate the costs in other Regions that are available. If the forecasted saving outweighs the cost of moving the component or workload, migrate to the new Region.
- **Review requirements for multi-Region deployments:** Analyze your business requirements and obligations (data privacy, security, or performance) to find out if there are any restrictions for you to not to use multiple Regions. If there are no obligations to restrict you to use single Region, then use multiple Regions.
- **Analyze required data transfer:** Consider data transfer costs when selecting Regions. Keep your data close to your customer and close to the resources. Select less costly AWS Regions where data flows and where there is minimal data transfer. Depending on your business requirements for data transfer, you can use Amazon CloudFront, AWS PrivateLink, AWS Direct Connect, and AWS Virtual Private Network to reduce your networking costs, improve performance, and enhance security.

Resources

Related documents:

- Accessing Reserved Instance recommendations
- Amazon EC2 pricing
- Instance purchasing options
- Region Table

Related videos:

- Save up to 90% and run production workloads on Spot

Related examples:

- Overview of Data Transfer Costs for Common Architectures
- Cost Considerations for Global Deployments
- What to Consider when Selecting a Region for your Workloads

COST07-BP03 Select third-party agreements with cost-efficient terms

Cost efficient agreements and terms ensure the cost of these services scales with the benefits they provide. Select agreements and pricing that scale when they provide additional benefits to your organization.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

There are multiple products on the market that can help you manage costs in your cloud environments. They may have some differences in terms of features that depend on customer requirements, such as some focusing on cost governance or cost visibility and others on cost optimization. One key factor for effective cost optimization and governance is using the right tool with necessary features and the right pricing model. These products have different pricing models. Some charge you a certain percentage of your monthly bill, while others charge a percentage of your realized savings. Ideally, you should pay only for what you need.

When you use third-party solutions or services in the cloud, it's important that the pricing structures are aligned to your desired outcomes. Pricing should scale with the outcomes and value it provides. For example, in software that takes a percentage of savings it provides, the more you save (outcome), the more it charges. License agreements where you pay more as your expenses increase might not always be in your best interest for optimizing costs. However, if the vendor offers clear benefits for all parts of your bill, this scaling fee might be justified.

For example, a solution that provides recommendations for Amazon EC2 and charges a percentage of your entire bill can become more expensive if you use other services that provide no benefit. Another example is a managed service that is charged at a percentage of the cost of managed resources. A larger instance size may not necessarily require more management effort, but can be charged more. Verify that these service pricing arrangements include a cost optimization program or features in their service to drive efficiency.

Customers may find these products on the market more advanced or easier to use. You need to consider the cost of these products and think about potential cost optimization outcomes in the long term.

Implementation steps

- **Analyze third-party agreements and terms:** Review the pricing in third party agreements. Perform modeling for different levels of your usage, and factor in new costs such as new service usage, or increases in current services due to workload growth. Decide if the additional costs provide the required benefits to your business.

Resources

Related documents:

- Accessing Reserved Instance recommendations
- Instance purchasing options

Related videos:

- Save up to 90% and run production workloads on Spot

COST07-BP04 Implement pricing models for all components of this workload

Permanently running resources should utilize reserved capacity such as Savings Plans or Reserved Instances. Short-term capacity is configured to use Spot Instances, or Spot Fleet. On-Demand Instances are only used for short-term workloads that cannot be interrupted and do not run long enough for reserved capacity, between 25% to 75% of the period, depending on the resource type.

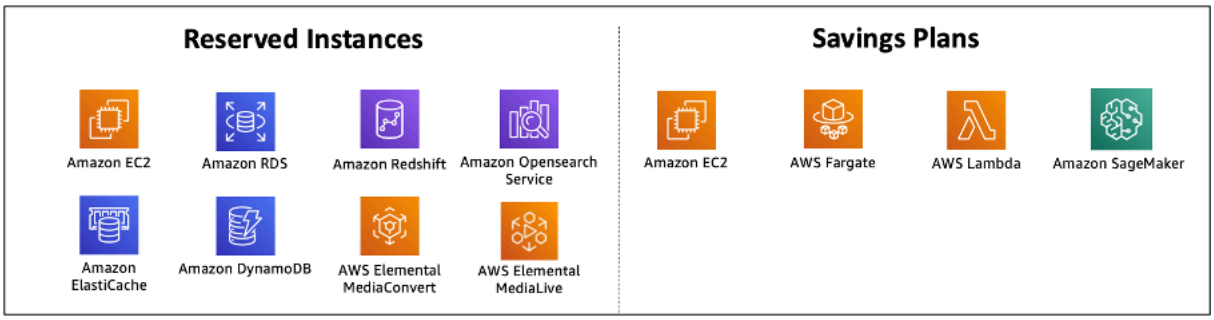
Level of risk exposed if this best practice is not established: Low

Implementation guidance

To improve cost efficiency, AWS provides multiple commitment recommendations based on your past usage. You can use these recommendations to understand what you can save, and how the commitment will be used. You can use these services as On-Demand, Spot, or make a commitment for a certain period of time and reduce your on-demand costs with Reserved Instances (RIs) and Savings Plans (SPs). You need to understand not only each workload components and multiple AWS services, but also commitment discounts, purchase options, and Spot Instances for these services to optimize your workload.

Consider the requirements of your workload's components, and understand the different pricing models for these services. Define the availability requirement of these components. Determine if there are multiple independent resources that perform the function in the workload, and what the workload requirements are over time. Compare the cost of the resources using the default On-Demand pricing model and other applicable models. Factor in any potential changes in resources or workload components.

For example, let's look at this Web Application Architecture on AWS. This sample workload consists of multiple AWS services, such as Amazon Route 53, AWS WAF, Amazon CloudFront, Amazon EC2 instances, Amazon RDS instances, Load Balancers, Amazon S3 storage, and Amazon Elastic File System (Amazon EFS). You need to review each of these services, and identify potential cost saving opportunities with different pricing models. Some of them may be eligible for RIs or SPs, while some of them may be available only on-demand. As the following image shows, some of the AWS services can be committed using RIs or SPs.



AWS services committed using Reserved Instances and Savings Plans

Implementation steps

- **Implement pricing models:** Using your analysis results, purchase Savings Plans, Reserved Instances, or implement Spot Instances. If it is your first commitment purchase, choose the top five or ten recommendations in the list, then monitor and analyze the results over the next month or two. AWS Cost Management Console guides you through the process. Review the RI or SP recommendations from the console, customize the recommendations (type, payment, and term), and review hourly commitment (for example $20 per hour), and then add to cart. Discounts apply automatically to eligible usage. Purchase a small amount of commitment discounts in regular cycles (for example every 2 weeks or monthly). Implement Spot Instances for workloads that can be interrupted or are stateless. Finally, select on-demand Amazon EC2 instances and allocate resources for the remaining requirements.
- **Workload review cycle:** Implement a review cycle for the workload that specifically analyzes pricing model coverage. Once the workload has the required coverage, purchase additional commitment discounts partially (every few months), or as your organization usage changes.

Resources

Related documents:

- Understanding your Savings Plans recommendations
- Accessing Reserved Instance recommendations
- How to Purchase Reserved Instances
- Instance purchasing options
- Spot Instances
- Reservation models for other AWS services

- Savings Plans Supported Services

Related videos:

- Save up to 90% and run production workloads on Spot

Related examples:

- What should you consider before purchasing Savings Plans?
- How can I use Cost Explorer to analyze my spending and usage?

COST07-BP05 Perform pricing model analysis at the management account level

Check billing and cost management tools and see recommended discounts with commitments and reservations to perform regular analysis at the management account level.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Performing regular cost modeling helps you implement opportunities to optimize across multiple workloads. For example, if multiple workloads use On-Demand Instances at an aggregate level, the risk of change is lower, and implementing a commitment-based discount can achieve a lower overall cost. It is recommended to perform analysis in regular cycles of two weeks to one month. This allows you to make small adjustment purchases, so the coverage of your pricing models continues to evolve with your changing workloads and their components.

Use the AWS Cost Explorer recommendations tool to find opportunities for commitment discounts in your management account. Recommendations at the management account level are calculated considering usage across all of the accounts in your AWS organization that have Reserve Instances (RI) or Savings Plans (SP). They're also calculated when discount sharing is activated to recommend a commitment that maximizes savings across accounts.

While purchasing at the management account level optimizes for max savings in many cases, there may be situations where you might consider purchasing SPs at the linked account level, like when you want the discounts to apply first to usage in that particular linked account. Member account recommendations are calculated at the individual account level, to maximize savings for each isolated account. If your account owns both RI and SP commitments, they will be applied in this order:

1. Zonal RI
2. Standard RI
3. Convertible RI
4. Instance Savings Plan
5. Compute Savings Plan

If you purchase an SP at the management account level, the savings will be applied based on highest to lowest discount percentage. SPs at the management account level look across all linked accounts and apply the savings wherever the discount will be the highest. If you wish to restrict where the savings are applied, you can purchase a Savings Plan at the linked account level and any time that account is running eligible compute services, the discount will be applied there first. When the account is not running eligible compute services, the discount will be shared across the other linked accounts under the same management account. Discount sharing is turned on by default, but can be turned off if needed.

In a Consolidated Billing Family, Savings Plans are applied first to the owner account's usage, and then to other accounts' usage. This occurs only if you have sharing enabled. Your Savings Plans are applied to your highest savings percentage first. If there are multiple usages with equal savings percentages, Savings Plans are applied to the first usage with the lowest Savings Plans rate. Savings Plans continue to apply until there are no more remaining uses or your commitment is exhausted. Any remaining usage is charged at the On-Demand rates. You can refresh Savings Plans Recommendations in AWS Cost Management to generate new Savings Plans Recommendations at any time.

After analyzing flexibility of instances, you can commit by following recommendations. Create cost modeling by analyzing the workload’s short-term costs with potential different resource options, analyzing AWS pricing models, and aligning them with your business requirements to find out total cost of ownership and cost optimization opportunities.

Implementation steps

Perform a commitment discount analysis: Use Cost Explorer in your account review the Savings Plans and Reserved Instance recommendations. Make sure you understand Saving Plan recommendations, and estimate your monthly spend and monthly savings. Review recommendations at the management account level, which are calculated considering usage across all of the member accounts in your AWS organization that have RI or Savings Plans discount sharing enabled for maximum savings across accounts. You can verify that you implemented the

correct recommendations with the required discounts and risk by following the Well-Architected labs.

Resources

Related documents:

- How does AWS pricing work?
- Instance purchasing options
- Saving Plan Overview
- Saving Plan recommendations
- Accessing Reserved Instance recommendations
- Understanding your Saving Plans recommendation
- How Savings Plans apply to your AWS usage
- Saving Plans with Consolidated Billing
- Turning on shared reserved instances and Savings Plans discounts

Related videos:

- Save up to 90% and run production workloads on Spot

Related examples:

- What should I consider before purchasing a Savings Plan?
- How can I use rolling Savings Plans to reduce commitment risk?
- When to Use Spot Instances

COST 8. How do you plan for data transfer charges?

Verify that you plan and monitor data transfer charges so that you can make architectural decisions to minimize costs. A small yet effective architectural change can drastically reduce your operational costs over time.

Best practices

- COST08-BP01 Perform data transfer modeling

- COST08-BP02 Select components to optimize data transfer cost
- COST08-BP03 Implement services to reduce data transfer costs

COST08-BP01 Perform data transfer modeling

Gather organization requirements and perform data transfer modeling of the workload and each of its components. This identifies the lowest cost point for its current data transfer requirements.

Level of risk exposed if this best practice is not established: High

Implementation guidance

When designing a solution in the cloud, data transfer fees are usually neglected due to habits of designing architecture using on-premises data centers or lack of knowledge. Data transfer charges in AWS are determined by the source, destination, and volume of traffic. Factoring in these fees during the design phase can lead to cost savings. Understanding where the data transfer occurs in your workload, the cost of the transfer, and its associated benefit is very important to accurately estimate total cost of ownership (TCO). This allows you to make an informed decision to modify or accept the architectural decision. For example, you may have a Multi-Availability Zone configuration where you replicate data between the Availability Zones.

You model the components of services which transfer the data in your workload, and decide that this is an acceptable cost (similar to paying for compute and storage in both Availability Zones) to achieve the required reliability and resilience. Model the costs over different usage levels. Workload usage can change over time, and different services may be more cost effective at different levels.

While modeling your data transfer, think about how much data is ingested and where that data comes from. Additionally, consider how much data is processed and how much storage or compute capacity is needed. During modeling, follow networking best practices for your workload architecture to optimize your potential data transfer costs.

The AWS Pricing Calculator can help you see estimated costs for specific AWS services and expected data transfer. If you have a workload already running (for test purposes or in a pre-production environment), use AWS Cost Explorer or AWS Cost and Usage Report (CUR) to understand and model your data transfer costs. Configure a proof of concept (PoC) or test your workload, and run a test with a realistic simulated load. You can model your costs at different workload demands.

Implementation steps

- **Identify requirements:** What is the primary goal and business requirements for the planned data transfer between source and destination? What is the expected business outcome at the end? Gather business requirements and define expected outcome.
- **Identify source and destination:** What is the data source and destination for the data transfer, such as within AWS Regions, to AWS services, or out to the internet?
  - Data transfer within an AWS Region
  - Data transfer between AWS Regions
  - Data transfer out to the internet
- **Identify data classifications:** What is the data classification for this data transfer? What kind of data is it? How big is the data? How frequently must data be transferred? Is data sensitive?
- **Identify AWS services or tools to use:** Which AWS services are used for this data transfer? Is it possible to use an already-provisioned service for another workload?
- **Calculate data transfer costs:** Use AWS Pricing the data transfer modeling you created previously to calculate the data transfer costs for the workload. Calculate the data transfer costs at different usage levels, for both increases and reductions in workload usage. Where there are multiple options for the workload architecture, calculate the cost for each option for comparison.
- **Link costs to outcomes:** For each data transfer cost incurred, specify the outcome that it achieves for the workload. If it is transfer between components, it may be for decoupling, if it is between Availability Zones it may be for redundancy.
- **Create data transfer modeling:** After gathering all information, create a conceptual base data transfer modeling for multiple use cases and different workloads.

Resources

Related documents:

- AWS caching solutions
- AWS Pricing
- Amazon EC2 Pricing
- Amazon VPC pricing
- Understanding data transfer charges

Related videos:

- Monitoring and Optimizing Your Data Transfer Costs
- S3 Transfer Acceleration

Related examples:

- Overview of Data Transfer Costs for Common Architectures
- AWS Prescriptive Guidance for Networking

COST08-BP02 Select components to optimize data transfer cost

All components are selected, and architecture is designed to reduce data transfer costs. This includes using components such as wide-area-network (WAN) optimization and Multi-Availability Zone (AZ) configurations

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Architecting for data transfer minimizes data transfer costs. This may involve using content delivery networks to locate data closer to users, or using dedicated network links from your premises to AWS. You can also use WAN optimization and application optimization to reduce the amount of data that is transferred between components.

When transferring data to or within the AWS Cloud, it is essential to know the destination based on varied use cases, the nature of the data, and the available network resources in order to select the right AWS services to optimize data transfer. AWS offers a range of data transfer services tailored for diverse data migration requirements. Select the right data storage and data transfer options based on the business needs within your organization.

When planning or reviewing your workload architecture, consider the following:

- **Use VPC endpoints within AWS:** VPC endpoints allow for private connections between your VPC and supported AWS services. This allows you to avoid using the public internet, which can lead to data transfer costs.
- **Use a NAT gateway:** Use a NAT gateway so that instances in a private subnet can connect to the internet or to the services outside your VPC. Check whether the resources behind the NAT gateway that send the most traffic are in the same Availability Zone as the NAT gateway. If they

are not, create new NAT gateways in the same Availability Zone as the resource to reduce cross-AZ data transfer charges.

- **Use AWS Direct Connect** Direct Connect bypasses the public internet and establishes a direct, private connection between your on-premises network and AWS. This can be more cost-effective and consistent than transferring large volumes of data over the internet.
- **Avoid transferring data across Regional boundaries:** Data transfers between AWS Regions (from one Region to another) typically incur charges. It should be a very thoughtful decision to pursue a multi-Region path. For more detail, see Multi-Region scenarios.
- **Monitor data transfer:** Use Amazon CloudWatch and VPC flow logs to capture details about your data transfer and network usage. Analyze captured network traffic information in your VPCs, such as IP address or range going to and from network interfaces.
- **Analyze your network usage:** Use metering and reporting tools such as AWS Cost Explorer, CUDOS Dashboards, or CloudWatch to understand data transfer cost of your workload.

Implementation steps

- **Select components for data transfer:** Using the data transfer modeling explained in COST08-BP01 Perform data transfer modeling, focus on where the largest data transfer costs are or where they would be if the workload usage changes. Look for alternative architectures or additional components that remove or reduce the need for data transfer (or lower its cost).

Resources

Related best practices:

- COST08-BP01 Perform data transfer modeling
- COST08-BP03 Implement services to reduce data transfer costs

Related documents:

- Cloud Data Migration
- AWS caching solutions
- Deliver content faster with Amazon CloudFront

Related examples:

- Overview of Data Transfer Costs for Common Architectures
- AWS Network Optimization Tips
- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format

COST08-BP03 Implement services to reduce data transfer costs

Implement services to reduce data transfer. For example, use edge locations or content delivery networks (CDN) to deliver content to end users, build caching layers in front of your application servers or databases, and use dedicated network connections instead of VPN for connectivity to the cloud.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

There are various AWS services that can help you to optimize your network data transfer usage. Depending on your workload components, type, and cloud architecture, these services can assist you in compression, caching, and sharing and distribution of your traffic on the cloud.

- Amazon CloudFront is a global content delivery network that delivers data with low latency and high transfer speeds. It caches data at edge locations across the world, which reduces the load on your resources. By using CloudFront, you can reduce the administrative effort in delivering content to large numbers of users globally with minimum latency. The security savings bundle can help you to save up to 30% on your CloudFront usage if you plan to grow your usage over time.
- AWS Direct Connect allows you to establish a dedicated network connection to AWS. This can reduce network costs, increase bandwidth, and provide a more consistent network experience than internet-based connections.
- Site-to-Site VPN allows you to establish a secure and private connection between your private network and the AWS global network. It is ideal for small offices or business partners because it provides simplified connectivity, and it is a fully managed and elastic service.
- VPC Endpoints allow connectivity between AWS services over private networking and can be used to reduce public data transfer and NAT gateway costs. Gateway VPC endpoints have no hourly charges, and support Amazon S3 and Amazon DynamoDB. Interface VPC endpoints are provided by AWS PrivateLink and have an hourly fee and per-GB usage cost.

- NAT gateways provide built-in scaling and management for reducing costs as opposed to a standalone NAT instance. Place NAT gateways in the same Availability Zones as high traffic instances and consider using VPC endpoints for the instances that need to access Amazon DynamoDB or Amazon S3 to reduce the data transfer and processing costs.
- Use AWS Snow Family devices which have computing resources to collect and process data at the edge. AWS Snow Family devices (Snowball Edge, Snowball Edge and Snowmobile) allow you to move petabytes of data to the AWS Cloud cost effectively and offline.

Implementation steps

- **Implement services:** Select applicable AWS network services based on your service workload type using the data transfer modeling and reviewing VPC Flow Logs. Look at where the largest costs and highest volume flows are. Review the AWS services and assess whether there is a service that reduces or removes the transfer, specifically networking and content delivery. Also look for caching services where there is repeated access to data or large amounts of data.

Resources

Related documents:

- AWS Direct Connect
- AWS Explore Our Products
- AWS caching solutions
- Amazon CloudFront
- AWS Snow Family
- Amazon CloudFront Security Savings Bundle

Related videos:

- Monitoring and Optimizing Your Data Transfer Costs
- AWS Cost Optimization Series: CloudFront
- How can I reduce data transfer charges for my NAT gateway?

Related examples:

- How-to chargeback shared services: An AWS Transit Gateway example
- Understand AWS data transfer details in depth from cost and usage report using Athena query and QuickSight
- Overview of Data Transfer Costs for Common Architectures
- Using AWS Cost Explorer to analyze data transfer costs
- Cost-Optimizing your AWS architectures by utilizing Amazon CloudFront features
- How can I reduce data transfer charges for my NAT gateway?

Manage demand and supply resources

Question

- COST 9. How do you manage demand, and supply resources?

COST 9. How do you manage demand, and supply resources?

For a workload that has balanced spend and performance, verify that everything you pay for is used and avoid significantly underutilizing instances. A skewed utilization metric in either direction has an adverse impact on your organization, in either operational costs (degraded performance due to over-utilization), or wasted AWS expenditures (due to over-provisioning).

Best practices

- COST09-BP01 Perform an analysis on the workload demand
- COST09-BP02 Implement a buffer or throttle to manage demand
- COST09-BP03 Supply resources dynamically

COST09-BP01 Perform an analysis on the workload demand

Analyze the demand of the workload over time. Verify that the analysis covers seasonal trends and accurately represents operating conditions over the full workload lifetime. Analysis effort should reflect the potential benefit, for example, time spent is proportional to the workload cost.

Level of risk exposed if this best practice is not established: High

Implementation guidance

Analyzing workload demand for cloud computing involves understanding the patterns and characteristics of computing tasks that are initiated in the cloud environment. This analysis helps users optimize resource allocation, manage costs, and verify that performance meets required levels.

Know the requirements of the workload. Your organization's requirements should indicate the workload response times for requests. The response time can be used to determine if the demand is managed, or if the supply of resources should change to meet the demand.

The analysis should include the predictability and repeatability of the demand, the rate of change in demand, and the amount of change in demand. Perform the analysis over a long enough period to incorporate any seasonal variance, such as end-of-month processing or holiday peaks.

Analysis effort should reflect the potential benefits of implementing scaling. Look at the expected total cost of the component and any increases or decreases in usage and cost over the workload's lifetime.

The following are some key aspects to consider when performing workload demand analysis for cloud computing:

1. **Resource utilization and performance metrics**: Analyze how AWS resources are being used over time. Determine peak and off-peak usage patterns to optimize resource allocation and scaling strategies. Monitor performance metrics such as response times, latency, throughput, and error rates. These metrics help assess the overall health and efficiency of the cloud infrastructure.
2. **User and application scaling behaviour**: Understand user behavior and how it affects workload demand. Examining the patterns of user traffic assists in enhancing the delivery of content and the responsiveness of applications. Analyze how workloads scale with increasing demand. Determine whether auto-scaling parameters are configured correctly and effectively for handling load fluctuations.
3. **Workload types**: Identify the different types of workloads running in the cloud, such as batch processing, real-time data processing, web applications, databases, or machine learning. Each type of workload may have different resource requirements and performance profiles.
4. **Service-level agreements (SLAs)**: Compare actual performance with SLAs to ensure compliance and identify areas that need improvement.

You can use Amazon CloudWatch to collect and track metrics, monitor log files, set alarms, and automatically react to changes in your AWS resources. You can also use Amazon CloudWatch to gain system-wide visibility into resource utilization, application performance, and operational health.

With AWS Trusted Advisor, you can provision your resources following best practices to improve system performance and reliability, increase security, and look for opportunities to save money. You can also turn off non-production instances and use Amazon CloudWatch and Auto Scaling to match increases or reductions in demand.

Finally, you can use AWS Cost Explorer or Quick with the AWS Cost and Usage Report (CUR) file or your application logs to perform advanced analysis of workload demand.

Overall, a comprehensive workload demand analysis allows organizations to make informed decisions about resource provisioning, scaling, and optimization, leading to better performance, cost efficiency, and user satisfaction.

Implementation steps

- **Analyze existing workload data:** Analyze data from the existing workload, previous versions of the workload, or predicted usage patterns. Use Amazon CloudWatch, log files and monitoring data to gain insight on how workload was used. Analyze a full cycle of the workload, and collect data for any seasonal changes such as end-of-month or end-of-year events. The effort reflected in the analysis should reflect the workload characteristics. The largest effort should be placed on high-value workloads that have the largest changes in demand. The least effort should be placed on low-value workloads that have minimal changes in demand.
- **Forecast outside influence:** Meet with team members from across the organization that can influence or change the demand in the workload. Common teams would be sales, marketing, or business development. Work with them to know the cycles they operate within, and if there are any events that would change the demand of the workload. Forecast the workload demand with this data.

Resources

Related documents:

- Amazon CloudWatch
- AWS Trusted Advisor

- AWS X-Ray
- AWS Auto Scaling
- AWS Instance Scheduler
- Getting started with Amazon SQS
- AWS Cost Explorer
- Quick

Related examples:

- Monitor, Track and Analyze for cost optimization
- Searching and analyzing logs in CloudWatch

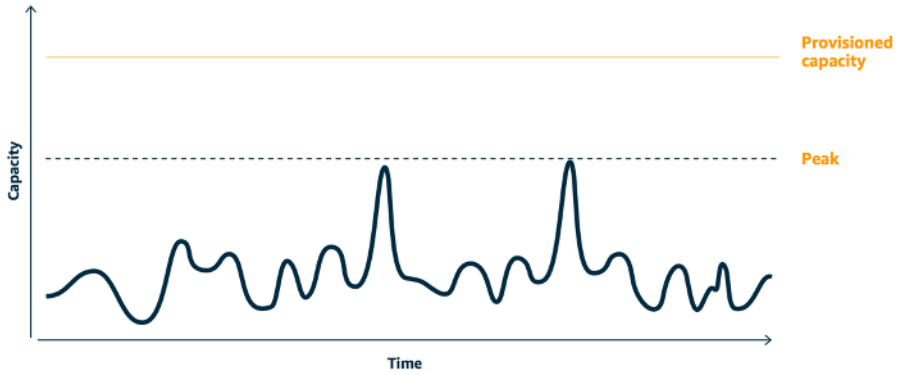
COST09-BP02 Implement a buffer or throttle to manage demand

Buffering and throttling modify the demand on your workload, smoothing out any peaks. Implement throttling when your clients perform retries. Implement buffering to store the request and defer processing until a later time. Verify that your throttles and buffers are designed so clients receive a response in the required time.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implementing a buffer or throttle is crucial in cloud computing in order to manage demand and reduce the provisioned capacity required for your workload. For optimal performance, it's essential to gauge the total demand, including peaks, the pace of change in requests, and the necessary response time. When clients have the ability to resend their requests, it becomes practical to apply throttling. Conversely, for clients lacking retry functionalities, the ideal approach is implementing a buffer solution. Such buffers streamline the influx of requests and optimize the interaction of applications with varied operational speeds.



Demand curve with two distinct peaks that require high provisioned capacity

Assume a workload with the demand curve shown in preceding image. This workload has two peaks, and to handle those peaks, the resource capacity as shown by orange line is provisioned. The resources and energy used for this workload are not indicated by the area under the demand curve, but the area under the provisioned capacity line, as provisioned capacity is needed to handle those two peaks. Flattening the workload demand curve can help you to reduce the provisioned capacity for a workload and reduce its environmental impact. To smooth out the peak, consider to implement throttling or buffering solution.

To understand them better, let's explore throttling and buffering.

Throttling: If the source of the demand has retry capability, then you can implement throttling. Throttling tells the source that if it cannot service the request at the current time, it should try again later. The source waits for a period of time, and then retries the request. Implementing throttling has the advantage of limiting the maximum amount of resources and costs of the workload. In AWS, you can use Amazon API Gateway to implement throttling.

Buffer based: A buffer-based approach uses *producers* (components that send messages to the queue), *consumers* (components that receive messages from the queue), and a *queue* (which holds messages) to store the messages. Messages are read by consumers and processed, allowing the messages to run at the rate that meets the consumers' business requirements. By using a buffer-centric methodology, messages from producers are housed in queues or streams, ready to be accessed by consumers at a pace that aligns with their operational demands.

In AWS, you can choose from multiple services to implement a buffering approach. Amazon Simple Queue Service(Amazon SQS) is a managed service that provides queues that allow a single consumer to read individual messages. Amazon Kinesis provides a stream that allows many consumers to read the same messages.

Buffering and throttling can smooth out any peaks by modifying the demand on your workload. Use throttling when clients retry actions and use buffering to hold the request and process it later. When working with a buffer-based approach, architect your workload to service the request in the required time, verify that you are able to handle duplicate requests for work. Analyze the overall demand, rate of change, and required response time to right size the throttle or buffer required.

Implementation steps

- **Analyze the client requirements:** Analyze the client requests to determine if they are capable of performing retries. For clients that cannot perform retries, buffers need to be implemented. Analyze the overall demand, rate of change, and required response time to determine the size of throttle or buffer required.
- **Implement a buffer or throttle:** Implement a buffer or throttle in the workload. A queue such as Amazon Simple Queue Service (Amazon SQS) can provide a buffer to your workload components. Amazon API Gateway can provide throttling for your workload components.

Resources

Related best practices:

- SUS02-BP06 Implement buffering or throttling to flatten the demand curve
- REL05-BP02 Throttle requests

Related documents:

- AWS Auto Scaling
- AWS Instance Scheduler
- Amazon API Gateway
- Amazon Simple Queue Service
- Getting started with Amazon SQS
- Amazon Kinesis

Related videos:

- Choosing the Right Messaging Service for Your Distributed App

Related examples:

- Managing and monitoring API throttling in your workloads
- Throttling a tiered, multi-tenant REST API at scale using API Gateway
- Enabling Tiering and Throttling in a Multi-Tenant Amazon EKS SaaS Solution Using Amazon API Gateway
- Application integration Using Queues and Messages

COST09-BP03 Supply resources dynamically

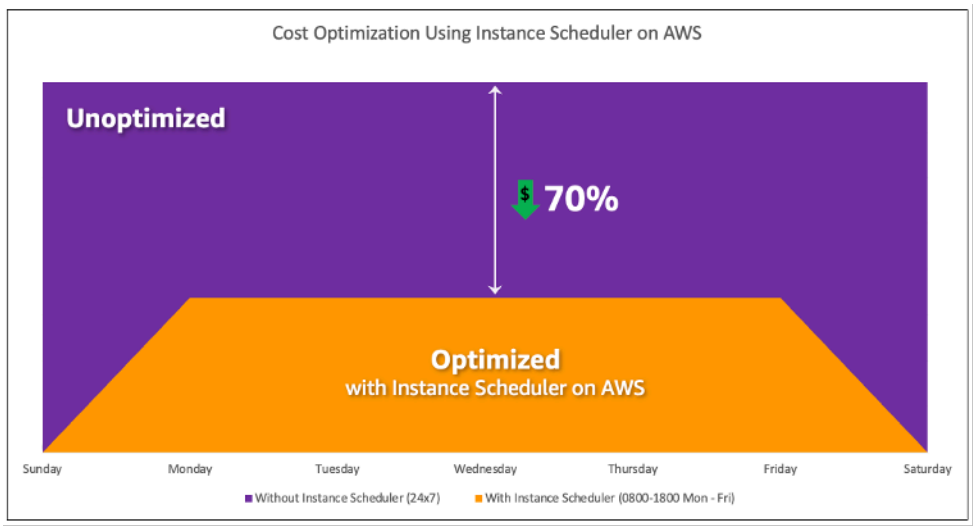
Resources are provisioned in a planned manner. This can be demand-based, such as through automatic scaling, or time-based, where demand is predictable and resources are provided based on time. These methods result in the least amount of over-provisioning or under-provisioning.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

There are several ways for AWS customers to increase the resources available to their applications and supply resources to meet the demand. One of these options is to use AWS Instance Scheduler, which automates the starting and stopping of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS) instances. The other option is to use AWS Auto Scaling, which allows you to automatically scale your computing resources based on the demand of your application or service. Supplying resources based on demand will allow you to pay for the resources you use only, reduce cost by launching resources when they are needed, and terminate them when they aren't.

AWS Instance Scheduler allows you to configure the stop and start of your Amazon EC2 and Amazon RDS instances at defined times so that you can meet the demand for the same resources within a consistent time pattern such as every day user access Amazon EC2 instances at eight in the morning that they don’t need after six at night. This solution helps reduce operational cost by stopping resources that are not in use and starting them when they are needed.



Cost optimization with AWS Instance Scheduler.

You can also easily configure schedules for your Amazon EC2 instances across your accounts and Regions with a simple user interface (UI) using AWS Systems Manager Quick Setup. You can schedule Amazon EC2 or Amazon RDS instances with AWS Instance Scheduler and you can stop and start existing instances. However, you cannot stop and start instances which are part of your Auto Scaling group (ASG) or that manage services such as Amazon Redshift or Amazon OpenSearch Service. Auto Scaling groups have their own scheduling for the instances in the group and these instances are created.

AWS Auto Scaling helps you adjust your capacity to maintain steady, predictable performance at the lowest possible cost to meet changing demand. It is a fully managed and free service to scale the capacity of your application that integrates with Amazon EC2 instances and Spot Fleets, Amazon ECS, Amazon DynamoDB, and Amazon Aurora. Auto Scaling provides automatic resource discovery to help find resources in your workload that can be configured, it has built-in scaling strategies to optimize performance, costs, or a balance between the two, and provides predictive scaling to assist with regularly occurring spikes.

There are multiple scaling options available to scale your Auto Scaling group:

- Maintain current instance levels at all times
- Scale manually
- Scale based on a schedule
- Scale based on demand
- Use predictive scaling

Auto Scaling policies differ and can be categorized as dynamic and scheduled scaling policies. Dynamic policies are manual or dynamic scaling which, scheduled or predictive scaling. You can use scaling policies for dynamic, scheduled, and predictive scaling. You can also use metrics and alarms from Amazon CloudWatch to trigger scaling events for your workload. We recommend you use launch templates, which allow you to access the latest features and improvements. Not all Auto Scaling features are available when you use launch configurations. For example, you cannot create an Auto Scaling group that launches both Spot and On-Demand Instances or that specifies multiple instance types. You must use a launch template to configure these features. When using launch templates, we recommended you version each one. With versioning of launch templates, you can create a subset of the full set of parameters. Then, you can reuse it to create other versions of the same launch template.

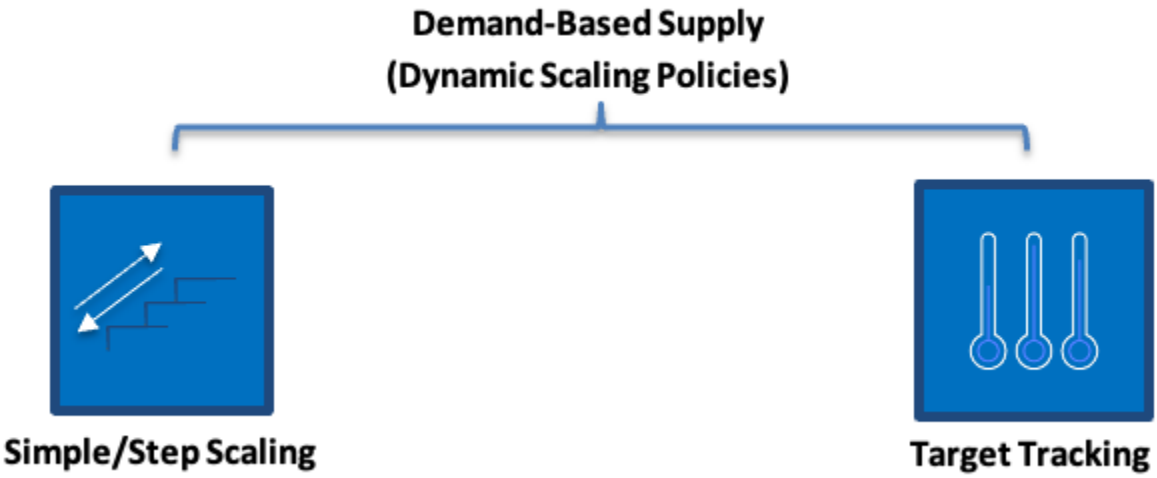
You can use AWS Auto Scaling or incorporate scaling in your code with AWS APIs or SDKs. This reduces your overall workload costs by removing the operational cost from manually making changes to your environment, and changes can be performed much faster. This also matches your workload resourcing to your demand at any time. In order to follow this best practice and supply resources dynamically for your organization, you should understand horizontal and vertical scaling in the AWS Cloud, as well as the nature of the applications running on Amazon EC2 instances. It is better for your Cloud Financial Management team to work with technical teams to follow this best practice.

Elastic Load Balancing (Elastic Load Balancing) helps you scale by distributing demand across multiple resources. By using ASG and Elastic Load Balancing, you can manage incoming requests by optimally routing traffic so that no one instance is overwhelmed in an Auto Scaling group. The requests would be distributed among all the targets of a target group in a round-robin fashion without consideration for capacity or utilization.

Typical metrics can be standard Amazon EC2 metrics, such as CPU utilization, network throughput, and Elastic Load Balancing observed request and response latency. When possible, you should use a metric that is indicative of customer experience, typically a custom metric that might originate from application code within your workload. To elaborate how to meet the demand dynamically in this document, we will group Auto Scaling into two categories as demand-based and time-based supply models and deep dive into each.

Demand-based supply: Take advantage of elasticity of the cloud to supply resources to meet changing demand by relying on near real-time demand state. For demand-based supply, use APIs or service features to programmatically vary the amount of cloud resources in your architecture. This allows you to scale components in your architecture and increase the number of resources

during demand spikes to maintain performance and decrease capacity when demand subsides to reduce costs.

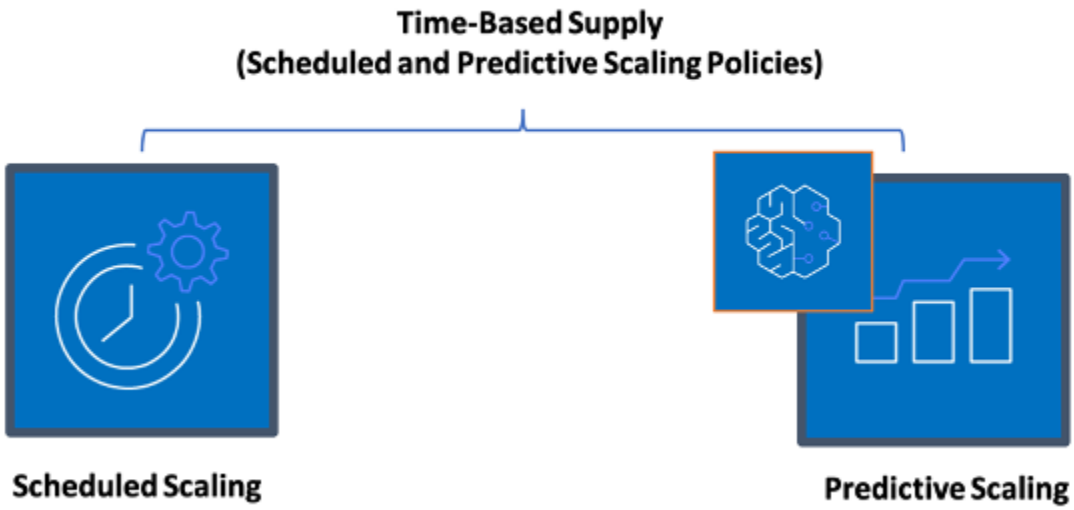


Demand-based dynamic scaling policies

- **Simple/Step Scaling:** Monitors metrics and adds/removes instances as per steps defined by the customers manually.
- **Target Tracking:** Thermostat-like control mechanism that automatically adds or removes instances to maintain metrics at a customer defined target.

When architecting with a demand-based approach keep in mind two key considerations. First, understand how quickly you must provision new resources. Second, understand that the size of margin between supply and demand will shift. You must be ready to cope with the rate of change in demand and also be ready for resource failures.

Time-based supply: A time-based approach aligns resource capacity to demand that is predictable or well-defined by time. This approach is typically not dependent upon utilization levels of the resources. A time-based approach ensures that resources are available at the specific time they are required and can be provided without any delays due to start-up procedures and system or consistency checks. Using a time-based approach, you can provide additional resources or increase capacity during busy periods.



Time-based scaling policies

You can use scheduled or predictive auto scaling to implement a time-based approach. Workloads can be scheduled to scale out or in at defined times (for example, the start of business hours), making resources available when users arrive or demand increases. Predictive scaling uses patterns to scale out while scheduled scaling uses pre-defined times to scale out. You can also use attribute-based instance type selection (ABS) strategy in Auto Scaling groups, which lets you express your instance requirements as a set of attributes, such as vCPU, memory, and storage. This also allows you to automatically use newer generation instance types when they are released and access a broader range of capacity with Amazon EC2 Spot Instances. Amazon EC2 Fleet and Amazon EC2 Auto Scaling select and launch instances that fit the specified attributes, removing the need to manually pick instance types.

You can also leverage the AWS APIs and SDKs and AWS CloudFormation to automatically provision and decommission entire environments as you need them. This approach is well suited for development or test environments that run only in defined business hours or periods of time. You can use APIs to scale the size of resources within an environment (vertical scaling). For example, you could scale up a production workload by changing the instance size or class. This can be achieved by stopping and starting the instance and selecting the different instance size or class. This technique can also be applied to other resources, such as Amazon EBS Elastic Volumes, which can be modified to increase size, adjust performance (IOPS) or change the volume type while in use.

When architecting with a time-based approach keep in mind two key considerations. First, how consistent is the usage pattern? Second, what is the impact if the pattern changes? You can increase the accuracy of predictions by monitoring your workloads and by using business intelligence. If you see significant changes in the usage pattern, you can adjust the times to ensure that coverage is provided.

Implementation steps

- **Configure scheduled scaling:** For predictable changes in demand, time-based scaling can provide the correct number of resources in a timely manner. It is also useful if resource creation and configuration is not fast enough to respond to changes on demand. Using the workload analysis configure scheduled scaling using AWS Auto Scaling. To configure time-based scheduling, you can use predictive scaling of scheduled scaling to increase the number of Amazon EC2 instances in your Auto Scaling groups in advance according to expected or predictable load changes.
- **Configure predictive scaling:** Predictive scaling allows you to increase the number of Amazon EC2 instances in your Auto Scaling group in advance of daily and weekly patterns in traffic flows. If you have regular traffic spikes and applications that take a long time to start, you should consider using predictive scaling. Predictive scaling can help you scale faster by initializing capacity before projected load compared to dynamic scaling alone, which is reactive in nature. For example, if users start using your workload with the start of the business hours and don't use after hours, then predictive scaling can add capacity before the business hours which eliminates delay of dynamic scaling to react to changing traffic.
- **Configure dynamic automatic scaling:** To configure scaling based on active workload metrics, use Auto Scaling. Use the analysis and configure Auto Scaling to launch on the correct resource levels, and verify that the workload scales in the required time. You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you to optimize your cost savings for Amazon EC2 instances and help you get the desired scale and performance for your application.

Resources

Related documents:

- AWS Auto Scaling

- AWS Instance Scheduler
- Scale the size of your Auto Scaling group
- Getting Started with Amazon EC2 Auto Scaling
- Getting started with Amazon SQS
- Scheduled Scaling for Amazon EC2 Auto Scaling
- Predictive scaling for Amazon EC2 Auto Scaling

Related videos:

- Target Tracking Scaling Policies for Auto Scaling
- AWS Instance Scheduler

Related examples:

- Attribute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- Optimizing Amazon Elastic Container Service for cost using scheduled scaling
- Predictive Scaling with Amazon EC2 Auto Scaling
- How do I use Instance Scheduler with CloudFormation to schedule Amazon EC2 instances?

Optimize over time

Questions

- COST 10. How do you evaluate new services?
- COST 11. How do you evaluate the cost of effort?

COST 10. How do you evaluate new services?

As AWS releases new services and features, it's a best practice to review your existing architectural decisions to verify they continue to be the most cost effective.

Best practices

- COST10-BP01 Develop a workload review process
- COST10-BP02 Review and analyze this workload regularly

COST10-BP01 Develop a workload review process

Develop a process that defines the criteria and process for workload review. The review effort should reflect potential benefit. For example, core workloads or workloads with a value of over ten percent of the bill are reviewed quarterly or every six months, while workloads below ten percent are reviewed annually.

Level of risk exposed if this best practice is not established: High

Implementation guidance

To have the most cost-efficient workload, you must regularly review the workload to know if there are opportunities to implement new services, features, and components. To achieve overall lower costs the process must be proportional to the potential amount of savings. For example, workloads that are 50% of your overall spend should be reviewed more regularly, and more thoroughly, than workloads that are five percent of your overall spend. Factor in any external factors or volatility. If the workload services a specific geography or market segment, and change in that area is predicted, more frequent reviews could lead to cost savings. Another factor in review is the effort to implement changes. If there are significant costs in testing and validating changes, reviews should be less frequent.

Factor in the long-term cost of maintaining outdated and legacy, components and resources and the inability to implement new features into them. The current cost of testing and validation may exceed the proposed benefit. However, over time, the cost of making the change may significantly increase as the gap between the workload and the current technologies increases, resulting in even larger costs. For example, the cost of moving to a new programming language may not currently be cost effective. However, in five years time, the cost of people skilled in that language may increase, and due to workload growth, you would be moving an even larger system to the new language, requiring even more effort than previously.

Break down your workload into components, assign the cost of the component (an estimate is sufficient), and then list the factors (for example, effort and external markets) next to each component. Use these indicators to determine a review frequency for each workload. For example, you may have webservers as a high cost, low change effort, and high external factors, resulting in high frequency of review. A central database may be medium cost, high change effort, and low external factors, resulting in a medium frequency of review.

Define a process to evaluate new services, design patterns, resource types, and configurations to optimize your workload cost as they become available. Similar to performance pillar review and

reliability pillar review processes, identify, validate, and prioritize optimization and improvement activities and issue remediation and incorporate this into your backlog.

Implementation steps

- **Define review frequency:** Define how frequently the workload and its components should be reviewed. Allocate time and resources to continual improvement and review frequency to improve the efficiency and optimization of your workload. This is a combination of factors and may differ from workload to workload within your organization and between components in the workload. Common factors include the importance to the organization measured in terms of revenue or brand, the total cost of running the workload (including operation and resource costs), the complexity of the workload, how easy is it to implement a change, any software licensing agreements, and if a change would incur significant increases in licensing costs due to punitive licensing. Components can be defined functionally or technically, such as web servers and databases, or compute and storage resources. Balance the factors accordingly and develop a period for the workload and its components. You may decide to review the full workload every 18 months, review the web servers every six months, the database every 12 months, compute and short-term storage every six months, and long-term storage every 12 months.
- **Define review thoroughness:** Define how much effort is spent on the review of the workload or workload components. Similar to the review frequency, this is a balance of multiple factors. Evaluate and prioritize opportunities for improvement to focus efforts where they provide the greatest benefits while estimating how much effort is required for these activities. If the expected outcomes do not satisfy the goals, and required effort costs more, then iterate using alternative courses of action. Your review processes should include dedicated time and resources to make continuous incremental improvements possible. As an example, you may decide to spend one week of analysis on the database component, one week of analysis for compute resources, and four hours for storage reviews.

Resources

Related documents:

- AWS News Blog
- Types of Cloud Computing
- What's New with AWS

Related examples:

- AWS Support Proactive Services
- Regular workload reviews for SAP workloads

COST10-BP02 Review and analyze this workload regularly

Existing workloads are regularly reviewed based on each defined process to find out if new services can be adopted, existing services can be replaced, or workloads can be re-architected.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

AWS is constantly adding new features so you can experiment and innovate faster with the latest technology. AWS What's New details how AWS is doing this and provides a quick overview of AWS services, features, and Regional expansion announcements as they are released. You can dive deeper into the launches that have been announced and use them for your review and analyze of your existing workloads. To realize the benefits of new AWS services and features, you review on your workloads and implement new services and features as required. This means you may need to replace existing services you use for your workload, or modernize your workload to adopt these new AWS services. For example, you might review your workloads and replace the messaging component with Amazon Simple Email Service. This removes the cost of operating and maintaining a fleet of instances, while providing all the functionality at a reduced cost.

To analyze your workload and highlight potential opportunities, you should consider not only new services but also new ways of building solutions. Review the This is My Architecture videos on AWS to learn about other customers' architecture designs, their challenges and their solutions. Check the All-In series to find out real world applications of AWS services and customer stories. You can also watch the Back to Basics video series that explains, examines, and breaks down basic cloud architecture pattern best practices. Another source is How to Build This videos, which are designed to assist people with big ideas on how to bring their minimum viable product (MVP) to life using AWS services. It is a way for builders from all over the world who have a strong idea to gain architectural guidance from experienced AWS Solutions Architects. Finally, you can review the Getting Started resource materials, which has step by step tutorials.

Before starting your review process, follow your business' requirements for the workload, security and data privacy requirements in order to use specific service or Region and performance requirements while following your agreed review process.

Implementation steps

- **Regularly review the workload:** Using your defined process, perform reviews with the frequency specified. Verify that you spend the correct amount of effort on each component. This process would be similar to the initial design process where you selected services for cost optimization. Analyze the services and the benefits they would bring, this time factor in the cost of making the change, not just the long-term benefits.
- **Implement new services:** If the outcome of the analysis is to implement changes, first perform a baseline of the workload to know the current cost for each output. Implement the changes, then perform an analysis to confirm the new cost for each output.

Resources

Related documents:

- AWS News Blog
- What's New with AWS
- AWS Documentation
- AWS Getting Started
- AWS General Resources

Related videos:

- AWS - This is My Architecture
- AWS - Back to Basics
- AWS - All-In series
- How to Build This

COST 11. How do you evaluate the cost of effort?

Best practices

- COST11-BP01 Perform automation for operations

COST11-BP01 Perform automation for operations

Evaluate the operational costs on the cloud, focusing on quantifying the time and effort savings in administrative tasks, deployments, mitigating the risk of human errors, compliance, and other

operations through automation. Assess the time and associated costs required for operational efforts and implement automation for administrative tasks to minimize manual effort wherever feasible.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Automating operations reduces the frequency of manual tasks, improves efficiency, and benefits customers by delivering a consistent and reliable experience when deploying, administering, or operating workloads. You can free up infrastructure resources from manual operational tasks and use them for higher value tasks and innovations, which improves business value. Enterprises require a proven, tested way to manage their workloads in the cloud. That solution must be secure, fast, and cost effective, with minimum risk and maximum reliability.

Start by prioritizing your operational activities based on required effort by looking at overall operations cost. For example, how long does it take to deploy new resources in the cloud, make optimization changes to existing ones, or implement necessary configurations? Look at the total cost of human actions by factoring in cost of operations and management. Prioritize automations for admin tasks to reduce the human effort.

Review effort should reflect the potential benefit. For example, examine time spent performing tasks manually as opposed to automatically. Prioritize automating repetitive, high value, time consuming and complex activities. Activities that pose a high value or high risk of human error are typically the better place to start automating as the risk often poses an unwanted additional operational cost (like operations team working extra hours).

Use automation tools like AWS Systems Manager or AWS Config to streamline operations, compliance, monitoring, lifecycle, and termination processes. With AWS services, tools, and third-party products, you can customize the automations you implement to meet your specific requirement. Following table shows some of the core operation functions and capabilities you can achieve with AWS services to automate administration and operation:

- AWS Audit Manager: Continually audit your AWS usage to simplify risk and compliance assessment
- AWS Backup: Centrally manage and automate data protection.
- AWS Config: Configure compute resources, asses, audit, evaluate configurations and resource inventory.
- AWS CloudFormation: Launch highly available resources with Infrastructure as Code.

- AWS CloudTrail: IT change management, compliance, and control.
- Amazon EventBridge Schedule events and trigger AWS Lambda to take action.
- AWS Lambda: Automate repetitive processes by triggering them with events or by running them on a fixed schedule with AWS EventBridge.
- AWS Systems Manager: Start and stop workloads, patch operating systems, automate configuration, and ongoing management.
- AWS Step Functions: Schedule jobs and automate workflows.
- AWS Service Catalog: Template consumption, infrastructure as code with compliance and control.

If you would like to adopt automations immediately by using AWS products and service and if don't have skills in your organization, reach out to AWS Managed Services (AMS), AWS Professional Services, or AWS Partners to increase adoption of automation and improve your operational excellence in the cloud.

AWS Managed Services (AMS) is a service that operates AWS infrastructure on behalf of enterprise customers and partners. It provides a secure and compliant environment that you can deploy your workloads onto. AMS uses enterprise cloud operating models with automation to allow you to meet your organization requirements, move into the cloud faster, and reduce your on-going management costs.

AWS Professional Services can also help you achieve your desired business outcomes and automate operations with AWS. They help customers to deploy automated, robust, agile IT operations, and governance capabilities optimized for the cloud. For detailed monitoring examples and recommended best practices, see Operational Excellence Pillar whitepaper.

Implementation steps

- **Build once and deploy many**: Use infrastructure-as-code such as CloudFormation, AWS SDK, or AWS CLI to deploy once and use many times for similar environments or for disaster recovery scenarios. Tag while deploying to track your consumption as defined in other best practices. Use AWS Launch Wizard to reduce the time to deploy many popular enterprise workloads. AWS Launch Wizard guides you through the sizing, configuration, and deployment of enterprise workloads following AWS best practices. You can also use the Service Catalog, which helps you create and manage infrastructure-as-code approved templates for use on AWS so anyone can discover approved, self-service cloud resources.
- **Automate continuous compliance:** Consider automating assessment and remediation of recorded configurations against predefined standards. When you combine AWS Organizations

with the capabilities of AWS Config and AWS CloudFormation, you can efficiently manage and automate configuration compliance at scale for hundreds of member accounts. You can review changes in configurations and relationships between AWS resources and dive into the history of a resource configuration.

- **Automate monitoring tasks** AWS provides various tools that you can use to monitor services. You can configure these tools to automate monitoring tasks. Create and implement a monitoring plan that collects monitoring data from all the parts in your workload so that you can more easily debug a multi-point failure if one occurs. For example, you can use the automated monitoring tools to observe Amazon EC2 and report back to you when something is wrong for system status checks, instance status checks, and Amazon CloudWatch alarms.
- **Automate maintenance and operations**: Run routine operations automatically without human intervention. Using AWS services and tools, you can choose which AWS automations to implement and customize for your specific requirements. For example, use EC2 Image Builder for building, testing, and deployment of virtual machine and container images for use on AWS or on-premises or patching your EC2 instances with AWS SSM. If your desired action cannot be done with AWS services or you need more complex actions with filtering resources, then automate your operations by using AWS Command Line Interface (AWS CLI) or AWS SDK tools. AWS CLI provides the ability to automate the entire process of controlling and managing AWS services with scripts without using the AWS Management Console. Select your preferred AWS SDKs to interact with AWS services. For other code examples, see AWS SDK Code examples repository.
- **Create a continual lifecycle with automations:** It is important that you establish and preserve mature lifecycle policies not only for regulations or redundancy but also for cost optimization. You can use AWS Backup to centrally manage and automate data protection of data stores, such as your buckets, volumes, databases, and file systems. You can also use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of EBS snapshots and EBS-backed AMIs.
- **Delete unnecessary resources:** It's quite common to accumulate unused resources in sandbox or development AWS accounts. Developers create and experiment with various services and resources as part of the normal development cycle, and then they don't delete those resources when they're no longer needed. Unused resources can incur unnecessary and sometimes high costs for the organization. Deleting these resources can reduce the costs of operating these environments. Make sure your data is not needed or backed up if you are not sure. You can use AWS CloudFormation to clean up deployed stacks, which automatically deletes most resources defined in the template. Alternatively, you can create an automation for the deletion of AWS resources using tools like aws-nuke.

Resources

Related documents:

- Modernizing operations in the AWS Cloud
- AWS Services for Automation
- Infrastructure and automation
- AWS Systems Manager Automation
- Automated and manual monitoring
- AWS automations for SAP administration and operations
- AWS Managed Services
- AWS Professional Services

Related videos:

- Automate Continuous Compliance at Scale in AWS
- AWS Backup Demo: Cross-Account & Cross-Region Backup
- Patching for your Amazon EC2 Instances

Related examples:

- Reinventing automated operations (Part I)
- Reinventing automated operations (Part II)
- Automate deletion of AWS resources by using aws-nuke
- Delete unused Amazon EBS volumes by using AWS Config and AWS SSM
- Automate continuous compliance at scale in AWS
- IT Automations with AWS Lambda

Sustainability

The Sustainability pillar includes understanding the impacts of the services used, quantifying impacts through the entire workload lifecycle, and applying design principles and best practices to reduce these impacts when building cloud workloads. You can find prescriptive guidance on implementation in the Sustainability Pillar whitepaper.

Best practice areas

- Region selection
- Alignment to demand
- Software and architecture
- Data
- Hardware and services
- Process and culture

Region selection

Question

- SUS 1 How do you select Regions for your workload?

SUS 1 How do you select Regions for your workload?

The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To effectively improve these KPIs, you should choose Regions for your workloads based on both business requirements and sustainability goals.

Best practices

- SUS01-BP01 Choose Region based on both business requirements and sustainability goals

SUS01-BP01 Choose Region based on both business requirements and sustainability goals

Choose a Region for your workload based on both your business requirements and sustainability goals to optimize its KPIs, including performance, cost, and carbon footprint.

Common anti-patterns:

- You select the workload's Region based on your own location.
- You consolidate all workload resources into one geographic location.

Benefits of establishing this best practice: Placing a workload close to Amazon renewable energy projects or Regions with low published carbon intensity can help to lower the carbon footprint of a cloud workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The AWS Cloud is a constantly expanding network of Regions and points of presence (PoP), with a global network infrastructure linking them together. The choice of Region for your workload significantly affects its KPIs, including performance, cost, and carbon footprint. To effectively improve these KPIs, you should choose Regions for your workload based on both your business requirements and sustainability goals.

Implementation steps

- **Shortlist potential Regions:** Follow these steps to assess and shortlist potential Regions for your workload based on your business requirements, including compliance, available features, cost, and latency:
  - Confirm that these Regions are compliant, based on your required local regulations (for example, data sovereignty).
  - Use the AWS Regional Services Lists to check if the Regions have the services and features you need to run your workload.
  - Calculate the cost of the workload on each Region using the AWS Pricing Calculator.
  - Test the network latency between your end user locations and each AWS Region.
- **Choose Regions:** Choose Regions near Amazon renewable energy projects and Regions where the grid has a published carbon intensity that is lower than other locations (or Regions).
  - Identify your relevant sustainability guidelines to track and compare year-to-year carbon emissions based on Greenhouse Gas Protocol (market-based and location based methods).
  - Choose region based on method you use to track carbon emissions. For more detail on choosing a Region based on your sustainability guidelines, see How to select a Region for your workload based on sustainability goals.

Resources

Related documents:

- Understanding your carbon emission estimations
- Amazon Around the Globe
- Renewable Energy Methodology
- What to Consider when Selecting a Region for your Workloads

Related videos:

- AWS re:Invent 2023 - Sustainability innovation in AWS Global Infrastructure
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure

Alignment to demand

Question

- SUS 2 How do you align cloud resources to your demand?

SUS 2 How do you align cloud resources to your demand?

The way users and applications consume your workloads and other resources can help you identify improvements to meet sustainability goals. Scale infrastructure to continually match demand and verify that you use only the minimum resources required to support your users. Align service levels to customer needs. Position resources to limit the network required for users and applications to consume them. Remove unused assets. Provide your team members with devices that support their needs and minimize their sustainability impact.

Best practices

- SUS02-BP01 Scale workload infrastructure dynamically
- SUS02-BP02 Align SLAs with sustainability goals
- SUS02-BP03 Stop the creation and maintenance of unused assets
- SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements
- SUS02-BP05 Optimize team member resources for activities performed
- SUS02-BP06 Implement buffering or throttling to flatten the demand curve

SUS02-BP01 Scale workload infrastructure dynamically

Use elasticity of the cloud and scale your infrastructure dynamically to match supply of cloud resources to demand and avoid overprovisioned capacity in your workload.

Common anti-patterns:

- You do not scale your infrastructure with user load.
- You manually scale your infrastructure all the time.
- You leave increased capacity after a scaling event instead of scaling back down.

Benefits of establishing this best practice: Configuring and testing workload elasticity help to efficiently match supply of cloud resources to demand and avoid overprovisioned capacity. You can take advantage of elasticity in the cloud to automatically scale capacity during and after demand spikes to make sure you are only using the right number of resources needed to meet your business requirements.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The cloud provides the flexibility to expand or reduce your resources dynamically through a variety of mechanisms to meet changes in demand. Optimally matching supply to demand delivers the lowest environmental impact for a workload.

Demand can be fixed or variable, requiring metrics and automation to make sure that management does not become burdensome. Applications can scale vertically (up or down) by modifying the instance size, horizontally (in or out) by modifying the number of instances, or a combination of both.

You can use a number of different approaches to match supply of resources with demand.

- **Target-tracking approach:** Monitor your scaling metric and automatically increase or decrease capacity as you need it.
- **Predictive scaling:** Scale in anticipation of daily and weekly trends.
- **Schedule-based approach:** Set your own scaling schedule according to predictable load changes.
- **Service scaling:** Pick services (like serverless) that are natively scaling by design or provide auto scaling as a feature.

Identify periods of low or no utilization and scale resources to remove excess capacity and improve efficiency.

Implementation steps

- Elasticity matches the supply of resources you have against the demand for those resources. Instances, containers, and functions provide mechanisms for elasticity, either in combination with automatic scaling or as a feature of the service. AWS provides a range of auto scaling mechanisms to ensure that workloads can scale down quickly and easily during periods of low user load. Here are some examples of auto scaling mechanisms:

Auto scaling mechanism	Where to use
Amazon EC2 Auto Scaling	Use to verify you have the correct number of Amazon EC2 instances available to handle the user load for your application.
Application Auto Scaling	Use to automatically scale the resources for individual AWS services beyond Amazon EC2, such as Lambda functions or Amazon Elastic Container Service (Amazon ECS) services.
Kubernetes Cluster Autoscaler	Use to automatically scale Kubernetes clusters on AWS.

- Scaling is often discussed related to compute services like Amazon EC2 instances or AWS Lambda functions. Consider the configuration of non-compute services like Amazon DynamoDB read and write capacity units or Amazon Kinesis Data Streams shards to match the demand.
- Verify that the metrics for scaling up or down are validated against the type of workload being deployed. If you are deploying a video transcoding application, 100% CPU utilization is expected and should not be your primary metric. You can use a customized metric (such as memory utilization) for your scaling policy if required. To choose the right metrics, consider the following guidance for Amazon EC2:
  - The metric should be a valid utilization metric and describe how busy an instance is.
  - The metric value must increase or decrease proportionally to the number of instances in the Auto Scaling group.
- Use dynamic scaling instead of manual scaling for your Auto Scaling group. We also recommend that you use target tracking scaling policies in your dynamic scaling.
- Verify that workload deployments can handle both scale-out and scale-in events. Create test scenarios for scale-in events to verify that the workload behaves as expected and does not affect

the user experience (like losing sticky sessions). You can use Activity history to verify a scaling activity for an Auto Scaling group.

- Evaluate your workload for predictable patterns and proactively scale as you anticipate predicted and planned changes in demand. With predictive scaling, you can eliminate the need to overprovision capacity. For more detail, see Predictive Scaling with Amazon EC2 Auto Scaling.

Resources

Related documents:

- Getting Started with Amazon EC2 Auto Scaling
- Predictive Scaling for EC2, Powered by Machine Learning
- Analyze user behavior using Amazon OpenSearch Service, Amazon Data Firehose and Kibana
- What is Amazon CloudWatch?
- Monitoring DB load with Performance Insights on Amazon RDS
- Introducing Native Support for Predictive Scaling with Amazon EC2 Auto Scaling
- Introducing Karpenter - An Open-Source, High-Performance Kubernetes Cluster Autoscaler
- Deep Dive on Amazon ECS Cluster Auto Scaling

Related videos:

- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment
- AWS re:Invent 2022 - Scaling containers from one user to millions
- AWS re:Invent 2023 - Scaling FM inference to hundreds of models with Amazon SageMaker AI
- AWS re:Invent 2023 - Harness the power of Karpenter to scale, optimize & upgrade Kubernetes

Related examples:

- Autoscaling

SUS02-BP02 Align SLAs with sustainability goals

Review and optimize workload service-level agreements (SLA) based on your sustainability goals to minimize the resources required to support your workload while continuing to meet business needs.

Common anti-patterns:

- Workload SLAs are unknown or ambiguous.
- You define your SLA just for availability and performance.
- You use the same design pattern (like Multi-AZ architecture) for all your workloads.

Benefits of establishing this best practice: Aligning SLAs with sustainability goals leads to optimal resource usage while meeting business needs.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

SLAs define the level of service expected from a cloud workload, such as response time, availability, and data retention. They influence the architecture, resource usage, and environmental impact of a cloud workload. At a regular cadence, review SLAs and make trade-offs that significantly reduce resource usage in exchange for acceptable decreases in service levels.

Implementation steps

- **Understand sustainability goals:** Identify sustainability goals in your organization, such as carbon reduction or improving resource utilization.
- **Review SLAs:** Evaluate your SLAs to assess if they support your business requirements. If you are exceeding SLAs, perform further review.
- **Understand trade-offs:** Understand the trade-offs across your workload's complexity (like high volume of concurrent users), performance (like latency), and sustainability impact (like required resources). Typically, prioritizing two of the factors comes at the expense of the third.
- **Adjust SLAs:** Adjust your SLAs by making trade-offs that significantly reduce sustainability impacts in exchange for acceptable decreases in service levels.
  - **Sustainability and reliability:** Highly available workloads tend to consume more resources.
  - **Sustainability and performance:** Using more resources to boost performance could have a higher environmental impact.

  - **Sustainability and security:** Overly secure workloads could have a higher environmental impact.
- **Define sustainability SLAs if possible:** Include sustainability SLAs for your workload. For example, define a minimum utilization level as a sustainability SLA for your compute instances.
- **Use efficient design patterns:** Use design patterns such as microservices on AWS that prioritize business-critical functions and allow lower service levels (such as response time or recovery time objectives) for non-critical functions.
- **Communicate and establish accountability:** Share the SLAs with all relevant stakeholders, including your development team and your customers. Use reporting to track and monitor the SLAs. Assign accountability to meet the sustainability targets for your SLAs.
- **Use incentives and rewards:** Use incentives and rewards to achieve or exceed SLAs aligned with sustainability goals.
- **Review and iterate:** Regularly review and adjust your SLAs to make sure they are aligned with evolving sustainability and performance goals.

Resources

Related documents:

- Understand resiliency patterns and trade-offs to architect efficiently in the cloud
- Importance of Service Level Agreement for SaaS Providers

Related videos:

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

SUS02-BP03 Stop the creation and maintenance of unused assets

Decommission unused assets in your workload to reduce the number of cloud resources required to support your demand and minimize waste.

Common anti-patterns:

- You do not analyze your application for assets that are redundant or no longer required.
- You do not remove assets that are redundant or no longer required.

Benefits of establishing this best practice: Removing unused assets frees resources and improves the overall efficiency of the workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Unused assets consume cloud resources like storage space and compute power. By identifying and eliminating these assets, you can free up these resources, resulting in a more efficient cloud architecture. Perform regular analysis on application assets such as pre-compiled reports, datasets, static images, and asset access patterns to identify redundancy, underutilization, and potential decommission targets. Remove those redundant assets to reduce the resource waste in your workload.

Implementation steps

- **Conduct an inventory:** Conduct a comprehensive inventory to identify all assets within your workload.
- **Analyze usage:** Use continuous monitoring to identify static assets that are no longer required.
- **Remove unused assets:** Develop a plan to remove assets that are no longer required.
  - Before removing any asset, evaluate the impact of removing it on the architecture.
  - Consolidate overlapping generated assets to remove redundant processing.
  - Update your applications to no longer produce and store assets that are not required.
- **Communicate with third parties:** Instruct third parties to stop producing and storing assets managed on your behalf that are no longer required. Ask to consolidate redundant assets.
- **Use lifecycle policies:** Use lifecycle policies to automatically delete unused assets.
  - You can use Amazon S3 Lifecycle to manage your objects throughout their lifecycle.
  - You can use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of Amazon EBS snapshots and Amazon EBS-backed AMIs.
- **Review and optimize:** Regularly review your workload to identify and remove any unused assets.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage
- How do I terminate active resources that I no longer need on my AWS account?

Related videos:

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Preserving and maximizing the value of digital media assets using Amazon S3
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

SUS02-BP04 Optimize geographic placement of workloads based on their networking requirements

Select cloud location and services for your workload that reduce the distance network traffic must travel and decrease the total network resources required to support your workload.

Common anti-patterns:

- You select the workload's Region based on your own location.
- You consolidate all workload resources into one geographic location.
- All traffic flows through your existing data centers.

Benefits of establishing this best practice: Placing a workload close to its users provides the lowest latency while decreasing data movement across the network and reducing environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The AWS Cloud infrastructure is built around location options such as Regions, Availability Zones, placement groups, and edge locations such as AWS Outposts and AWS Local Zones. These location options are responsible for maintaining connectivity between application components, cloud services, edge networks and on-premises data centers.

Analyze the network access patterns in your workload to identify how to use these cloud location options and reduce the distance network traffic must travel.

Implementation steps

- Analyze network access patterns in your workload to identify how users use your application.
  - Use monitoring tools, such as Amazon CloudWatch and AWS CloudTrail, to gather data on network activities.
  - Analyze the data to identify the network access pattern.
- Select the Regions for your workload deployment based on the following key elements:
  - **Your Sustainability goal:** as explained in Region selection.
  - **Where your data is located:** For data-heavy applications (such as big data and machine learning), application code should run as close to the data as possible.
  - **Where your users are located:** For user-facing applications, choose a Region (or Regions) close to your workload's users.
  - **Other constraints:** Consider constraints such as cost and compliance as explained in What to Consider when Selecting a Region for your Workloads.
- Use local caching or AWS Caching Solutions for frequently used assets to improve performance, reduce data movement, and lower environmental impact.

Service	When to use
Amazon CloudFront	Use to cache static content such as images, scripts, and videos, as well as dynamic content such as API responses or web applications.
Amazon ElastiCache	Use to cache content for web applications.
DynamoDB Accelerator	Use to add in-memory acceleration to your DynamoDB tables.

- Use services that can help you run code closer to users of your workload:

Service	When to use
Lambda@Edge	Use for compute-heavy operations that are initiated when objects are not in the cache.
Amazon CloudFront Functions	Use for simple use cases like HTTP(s) request or response manipulations that can be initiated by short-lived functions.
AWS IoT Greengrass	Use to run local compute, messaging, and data caching for connected devices.

- Use connection pooling to allow for connection reuse and reduce required resources.
- Use distributed data stores that don't rely on persistent connections and synchronous updates for consistency to serve regional populations.
- Replace pre-provisioned static network capacity with shared dynamic capacity, and share the sustainability impact of network capacity with other subscribers.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking
- Amazon ElastiCache Documentation
- What is Amazon CloudFront?
- Amazon CloudFront Key Features
- AWS Global Infrastructure
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload
- Placement groups
- AWS Local Zones
- AWS Outposts

Related videos:

- Demystifying data transfer on AWS

- Scaling network performance on next-gen Amazon EC2 instances
- AWS Local Zones Explainer Video
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2023 - A migration strategy for edge and on-premises workloads
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises
- AWS re:Invent 2020 - AWS Wavelength: Run apps with ultra-low latency at 5G edge
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator
- AWS re:Invent 2022 - Build your global wide area network using AWS
- AWS re:Invent 2020: Global traffic management with Amazon Route 53

Related examples:

- AWS Networking Workshops
- Architecting for sustainability - Minimize data movement across networks

SUS02-BP05 Optimize team member resources for activities performed

Optimize resources provided to team members to minimize the environmental sustainability impact while supporting their needs.

Common anti-patterns:

- You ignore the impact of devices used by your team members on the overall efficiency of your cloud application.
- You manually manage and update resources used by team members.

Benefits of establishing this best practice: Optimizing team member resources improves the overall efficiency of cloud-enabled applications.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Understand the resources your team members use to consume your services, their expected lifecycle, and the financial and sustainability impact. Implement strategies to optimize these resources. For example, perform complex operations, such as rendering and compilation, on highly utilized scalable infrastructure instead of on underutilized high-powered single-user systems.

Implementation steps

- **Use energy-efficient workstations:** Provide team members with energy-efficient workstations and peripherals. Use efficient power management features (like low power mode) in these devices to reduce their energy usage
- **use virtualization:** Use virtual desktops and application streaming to limit upgrade and device requirements.
- **Encourage remote collaboration:** Encourage team members to use remote collaboration tools such as Amazon Chime or AWS Wickr to reduce the need for travel and associated carbon emissions.
- **Use energy-efficient software:** Provide team members with energy-efficient software by removing or turning off unnecessary features and processes.
- **Manage lifecycles:** Evaluate the impact of processes and systems on your device lifecycle, and select solutions that minimize the requirement for device replacement while satisfying business requirements. Regularly maintain and update workstations or software to maintain and improve efficiency.
- **Remote device management:** Implement remote management for devices to reduce required business travel.
  - AWS Systems Manager Fleet Manager is a unified user interface (UI) experience that helps you remotely manage your nodes running on AWS or on premises.

Resources

Related documents:

- What is Amazon WorkSpaces?
- Cost Optimizer for Amazon WorkSpaces
- Amazon AppStream 2.0 Documentation
- NICE DCV

Related videos:

- Managing cost for Amazon WorkSpaces on AWS

SUS02-BP06 Implement buffering or throttling to flatten the demand curve

Buffering and throttling flatten the demand curve and reduce the provisioned capacity required for your workload.

Common anti-patterns:

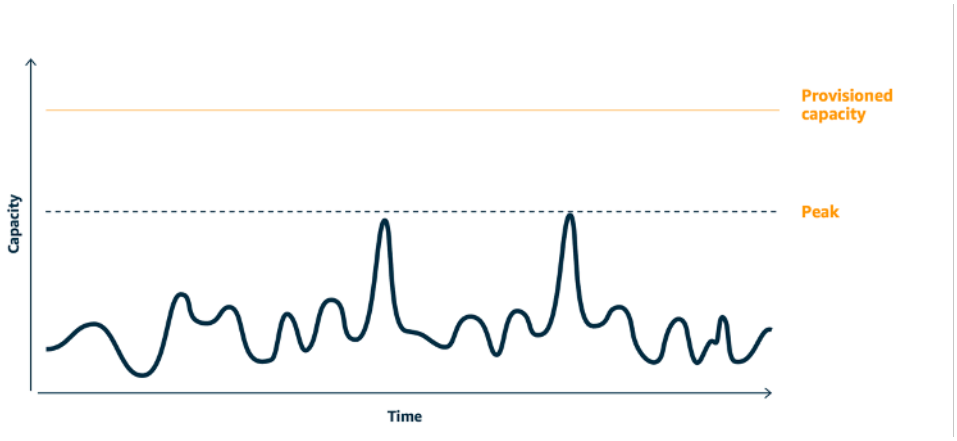
- You process the client requests immediately while it is not needed.
- You do not analyze the requirements for client requests.

Benefits of establishing this best practice: Flattening the demand curve reduce the required provisioned capacity for the workload. Reducing the provisioned capacity means less energy consumption and less environmental impact.

Level of risk exposed if this best practice is not established: Low

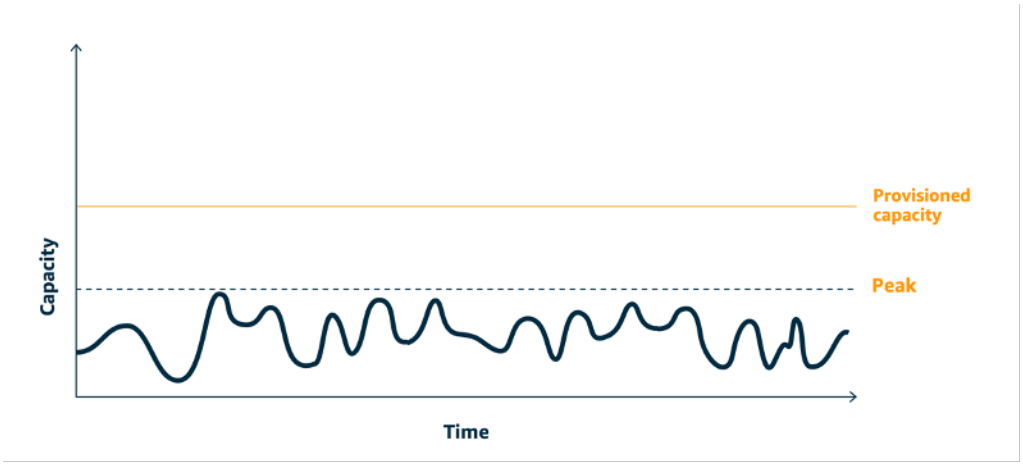
Implementation guidance

Flattening the workload demand curve can help you to reduce the provisioned capacity for a workload and reduce its environmental impact. Assume a workload with the demand curve shown in below figure. This workload has two peaks, and to handle those peaks, the resource capacity as shown by orange line is provisioned. The resources and energy used for this workload is not indicated by the area under the demand curve, but the area under the provisioned capacity line, as provisioned capacity is needed to handle those two peaks.



Demand curve with two distinct peaks that require high provisioned capacity.

You can use buffering or throttling to modify the demand curve and smooth out the peaks, which means less provisioned capacity and less energy consumed. Implement throttling when your clients can perform retries. Implement buffering to store the request and defer processing until a later time.



Throttling's effect on the demand curve and provisioned capacity.

Implementation steps

- Analyze the client requests to determine how to respond to them. Questions to consider include:
  - Can this request be processed asynchronously?
  - Does the client have retry capability?
- If the client has retry capability, then you can implement throttling, which tells the source that if it cannot service the request at the current time, it should try again later.
  - You can use Amazon API Gateway to implement throttling.
- For clients that cannot perform retries, a buffer needs to be implemented to flatten the demand curve. A buffer defers request processing, allowing applications that run at different rates to communicate effectively. A buffer-based approach uses a queue or a stream to accept messages from producers. Messages are read by consumers and processed, allowing the messages to run at the rate that meets the consumers' business requirements.
  - Amazon Simple Queue Service (Amazon SQS) is a managed service that provides queues that allow a single consumer to read individual messages.
  - Amazon Kinesis provides a stream that allows many consumers to read the same messages.
- Analyze the overall demand, rate of change, and required response time to right size the throttle or buffer required.

Resources

Related documents:

- Getting started with Amazon SQS
- Application integration Using Queues and Messages
- Managing and monitoring API throttling in your workloads
- Throttling a tiered, multi-tenant REST API at scale using API Gateway
- Application integration Using Queues and Messages

Related videos:

- AWS re:Invent 2022 - Application integration patterns for microservices
- AWS re:Invent 2023 - Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

Software and architecture

Question

- SUS 3 How do you take advantage of software and architecture patterns to support your sustainability goals?

SUS 3 How do you take advantage of software and architecture patterns to support your sustainability goals?

Implement patterns for performing load smoothing and maintaining consistent high utilization of deployed resources to minimize the resources consumed. Components might become idle from lack of use because of changes in user behavior over time. Revise patterns and architecture to consolidate under-utilized components to increase overall utilization. Retire components that are no longer required. Understand the performance of your workload components, and optimize the components that consume the most resources. Be aware of the devices that your customers use to access your services, and implement patterns to minimize the need for device upgrades.

Best practices

- SUS03-BP01 Optimize software and architecture for asynchronous and scheduled jobs

- SUS03-BP02 Remove or refactor workload components with low or no use
- SUS03-BP03 Optimize areas of code that consume the most time or resources
- SUS03-BP04 Optimize impact on devices and equipment
- SUS03-BP05 Use software patterns and architectures that best support data access and storage patterns

SUS03-BP01 Optimize software and architecture for asynchronous and scheduled jobs

Use efficient software and architecture patterns such as queue-driven to maintain consistent high utilization of deployed resources.

Common anti-patterns:

- You overprovision the resources in your cloud workload to meet unforeseen spikes in demand.
- Your architecture does not decouple senders and receivers of asynchronous messages by a messaging component.

Benefits of establishing this best practice:

- Efficient software and architecture patterns minimize the unused resources in your workload and improve the overall efficiency.
- You can scale the processing independently of the receiving of asynchronous messages.
- Through a messaging component, you have relaxed availability requirements that you can meet with fewer resources.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Use efficient architecture patterns such as event-driven architecture that result in even utilization of components and minimize overprovisioning in your workload. Using efficient architecture patterns minimizes idle resources from lack of use due to changes in demand over time.

Understand the requirements of your workload components and adopt architecture patterns that increase overall utilization of resources. Retire components that are no longer required.

Implementation steps

- Analyze the demand for your workload to determine how to respond to those.
- For requests or jobs that don't require synchronous responses, use queue-driven architectures and auto scaling workers to maximize utilization. Here are some examples of when you might consider queue-driven architecture:

Queuing mechanism	Description
AWS Batch job queues	AWS Batch jobs are submitted to a job queue where they reside until they can be scheduled to run in a compute environment.
Amazon Simple Queue Service and Amazon EC2 Spot Instances	Pairing Amazon SQS and Spot Instances to build fault tolerant and efficient architecture.

- For requests or jobs that can be processed anytime, use scheduling mechanisms to process jobs in batch for more efficiency. Here are some examples of scheduling mechanisms on AWS:

Scheduling mechanism	Description
Amazon EventBridge Scheduler	A capability from Amazon EventBridge that allows you to create, run, and manage scheduled tasks at scale.
AWS Glue time-based schedule	Define a time-based schedule for your crawlers and jobs in AWS Glue.
Amazon Elastic Container Service (Amazon ECS) scheduled tasks	Amazon ECS supports creating scheduled tasks. Scheduled tasks use Amazon EventBridge rules to run tasks either on a schedule or in a response to an EventBridge event.
Instance Scheduler	Configure start and stop schedules for your Amazon EC2 and Amazon Relational Database Service instances.

- If you use polling and webhooks mechanisms in your architecture, replace those with events. Use event-driven architectures to build highly efficient workloads.

- Leverage serverless on AWS to eliminate over-provisioned infrastructure.
- Right size individual components of your architecture to prevent idling resources waiting for input.
  - You can use the Rightsizing Recommendations in AWS Cost Explorer or AWS Compute Optimizer to identify rightsizing opportunities.
  - For more detail, see Right Sizing: Provisioning Instances to Match Workloads.

Resources

Related documents:

- What is Amazon Simple Queue Service?
- What is Amazon MQ?
- Scaling based on Amazon SQS
- What is AWS Step Functions?
- What is AWS Lambda?
- Using AWS Lambda with Amazon SQS
- What is Amazon EventBridge?
- Managing Asynchronous Workflows with a REST API

Related videos:

- AWS re:Invent 2023 - Navigating the journey to serverless event-driven architecture
- AWS re:Invent 2023 - Using serverless for event-driven architecture & domain-driven design
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- Asynchronous Message Patterns | AWS Events

Related examples:

- Event-driven architecture with AWS Graviton Processors and Amazon EC2 Spot Instances

SUS03-BP02 Remove or refactor workload components with low or no use

Remove components that are unused and no longer required, and refactor components with little utilization to minimize waste in your workload.

Common anti-patterns:

- You do not regularly check the utilization level of individual components of your workload.
- You do not check and analyze recommendations from AWS rightsizing tools such as AWS Compute Optimizer.

Benefits of establishing this best practice: Removing unused components minimizes waste and improves the overall efficiency of your cloud workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Unused or underutilized components in a cloud workload consume unnecessary compute, storage or network resources. Remove or refactor these components to directly reduce waste and improve the overall efficiency of a cloud workload. This is an iterative improvement process which can be initiated by changes in demand or the release of a new cloud service. For example, a significant drop in AWS Lambda function run time can be indicate a need to lower the memory size. Also, as AWS releases new services and features, the optimal services and architecture for your workload may change.

Continually monitor workload activity and look for opportunities to improve the utilization level of individual components. By removing idle components and performing rightsizing activities, you meet your business requirements with the fewest cloud resources.

Implementation steps

- **Inventory your AWS resourceds:** Create an inventory of your AWS resources. In AWS, you can turn on AWS Resource Explorer to explore and organize your AWS resources. For more details, see AWS re:Invent 2022 - How to manage resources and applications at scale on AWS.
- **Monitor utilization:** Monitor and capture the utilization metrics for critical components of your workload (like CPU utilization, memory utilization, or network throughput in Amazon CloudWatch metrics).

- **Identify unused components:** Identify unused or under-utilized components in your architecture.
  - For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer at regular intervals to identify idle, unused, or underutilized components.
  - For ephemeral workloads, evaluate utilization metrics to identify idle, unused, or underutilized components.
- **Remove unused components:** Retire components and associated assets (like Amazon ECR images) that are no longer needed.
  - Automated Cleanup of Unused Images in Amazon ECR
  - Delete unused Amazon Elastic Block Store (Amazon EBS) volumes by using AWS Config and AWS Systems Manager
- **Refactor underutilized components:** Refactor or consolidate underutilized components with other resources to improve utilization efficiency. For example, you can provision multiple small databases on a single Amazon RDS database instance instead of running databases on individual underutilized instances.
- **Evaluate improvements:** Understand the resources provisioned by your workload to complete a unit of work. Use this information to evaluate improvements achieved by removing or refactoring components.
  - Measure and track cloud efficiency with sustainability proxy metrics, Part I: What are proxy metrics?
  - Measure and track cloud efficiency with sustainability proxy metrics, Part II: Establish a metrics pipeline

Resources

Related documents:

- AWS Trusted Advisor
- What is Amazon CloudWatch?
- Right Sizing: Provisioning Instances to Match Workloads
- Optimizing your cost with Rightsizing Recommendations

Related videos:

- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three

SUS03-BP03 Optimize areas of code that consume the most time or resources

Optimize your code that runs within different components of your architecture to minimize resource usage while maximizing performance.

Common anti-patterns:

- You ignore optimizing your code for resource usage.
- You usually respond to performance issues by increasing the resources.
- Your code review and development process does not track performance changes.

Benefits of establishing this best practice: Using efficient code minimizes resource usage and improves performance.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

It is crucial to examine every functional area, including the code for a cloud architected application, to optimize its resource usage and performance. Continually monitor your workload's performance in build environments and production and identify opportunities to improve code snippets that have particularly high resource usage. Adopt a regular review process to identify bugs or anti-patterns within your code that use resources inefficiently. Leverage simple and efficient algorithms that produce the same results for your use case.

Implementation steps

- **Use efficient programming language:** Use an efficient operating system and programming language for the workload. For details on energy efficient programming languages (including Rust), see Sustainability with Rust.
- **Use an AI coding companion:** Consider using an AI coding companion such as Amazon Q Developer to efficiently write code.
- **Automate code reviews:** While developing your workloads, adopt an automated code review process to improve quality and identify bugs and anti-patterns.
  - Automate code reviews with Amazon CodeGuru Reviewer
  - Detecting concurrency bugs with Amazon CodeGuru
  - Raising code quality for Python applications using Amazon CodeGuru

- **Use a code profiler:** Use a code profiler to identify the areas of code that use the most time or resources as targets for optimization.
  - Reducing your organization's carbon footprint with Amazon CodeGuru Profiler
  - Understanding memory usage in your Java application with Amazon CodeGuru Profiler
  - Improving customer experience and reducing cost with Amazon CodeGuru Profiler
- **Monitor and optimize:** Use continuous monitoring resources to identify components with high resource requirements or suboptimal configuration.
  - Replace computationally intensive algorithms with simpler and more efficient version that produce the same result.
  - Remove unnecessary code such as sorting and formatting.
- **Use code refactoring or transformation:** Explore the possibility of Amazon Q code transformation for application maintenance and upgrades.
  - Upgrade language versions with Amazon Q Code Transformation
  - AWS re:Invent 2023 - Automate app upgrades & maintenance using Amazon Q Code Transformation

Resources

Related documents:

- What is Amazon CodeGuru Profiler?
- FPGA instances
- The AWS SDKs on Tools to Build on AWS

Related videos:

- Improve Code Efficiency Using Amazon CodeGuru Profiler
- Automate Code Reviews and Application Performance Recommendations with Amazon CodeGuru

SUS03-BP04 Optimize impact on devices and equipment

Understand the devices and equipment used in your architecture and use strategies to reduce their usage. This can minimize the overall environmental impact of your cloud workload.

Common anti-patterns:

- You ignore the environmental impact of devices used by your customers.
- You manually manage and update resources used by customers.

Benefits of establishing this best practice: Implementing software patterns and features that are optimized for customer device can reduce the overall environmental impact of cloud workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Implementing software patterns and features that are optimized for customer devices can reduce the environmental impact in several ways:

- Implementing new features that are backward compatible can reduce the number of hardware replacements.
- Optimizing an application to run efficiently on devices can help to reduce their energy consumption and extend their battery life (if they are powered by battery).
- Optimizing an application for devices can also reduce the data transfer over the network.

Understand the devices and equipment used in your architecture, their expected lifecycle, and the impact of replacing those components. Implement software patterns and features that can help to minimize the device energy consumption, the need for customers to replace the device and also upgrade it manually.

Implementation steps

- **Conduct an inventory:** Inventory the devices used in your architecture. Devices can be mobile, tablet, IOT devices, smart light, or even smart devices in a factory.
- **Use energy-efficient devices:** Consider using energy-efficient devices in your architecture. Use power management configurations on devices to enter low power mode when not in use.
- **Run efficient applications:** Optimize the application running on the devices:
  - Use strategies such as running tasks in the background to reduce their energy consumption.
  - Account for network bandwidth and latency when building payloads, and implement capabilities that help your applications work well on low bandwidth, high latency links.

  - Convert payloads and files into optimized formats required by devices. For example, you can use Amazon Elastic Transcoder or AWS Elemental MediaConvert to convert large, high quality digital media files into formats that users can play back on mobile devices, tablets, web browsers, and connected televisions.
  - Perform computationally intense activities server-side (such as image rendering), or use application streaming to improve the user experience on older devices.
  - Segment and paginate output, especially for interactive sessions, to manage payloads and limit local storage requirements.
- **Engage suppliers:** Work with device suppliers who use sustainable materials and provide transparency in their supply chains and environmental certifications.
- **Use over-the-air (OTA) updates:** Use automated over-the-air (OTA) mechanism to deploy updates to one or more devices.
  - You can use a CI/CD pipeline to update mobile applications.
  - You can use AWS IoT Device Management to remotely manage connected devices at scale.
- **Use managed device farms:** To test new features and updates, use managed device farms with representative sets of hardware and iterate development to maximize the devices supported. For more details, see SUS06-BP05 Use managed device farms for testing.
- **Continue to monitor and improve:** Track the energy usage of devices to identify areas for improvement. Use new technologies or best practices to enhance environmental impacts of these devices.

Resources

Related documents:

- What is AWS Device Farm?
- WorkSpaces Applications Documentation
- NICE DCV
- OTA tutorial for updating firmware on devices running FreeRTOS
- Optimizing Your IoT Devices for Environmental Sustainability

Related videos:

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm

SUS03-BP05 Use software patterns and architectures that best support data access and storage patterns

Understand how data is used within your workload, consumed by your users, transferred, and stored. Use software patterns and architectures that best support data access and storage to minimize the compute, networking, and storage resources required to support the workload.

Common anti-patterns:

- You assume that all workloads have similar data storage and access patterns.
- You only use one tier of storage, assuming all workloads fit within that tier.
- You assume that data access patterns will stay consistent over time.
- Your architecture supports a potential high data access burst, which results in the resources remaining idle most of the time.

Benefits of establishing this best practice: Selecting and optimizing your architecture based on data access and storage patterns will help decrease development complexity and increase overall utilization. Understanding when to use global tables, data partitioning, and caching will help you decrease operational overhead and scale based on your workload needs.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

To improve long-term workload sustainability, use architecture patterns that support data access and storage characteristics for your workload. These patterns help you efficiently retrieve and process data. For example, you can use modern data architecture on AWS with purpose-built services optimized for your unique analytics use cases. These architecture patterns allow for efficient data processing and reduce the resource usage.

Implementation steps

- **Understand data characteristics:** Analyze your data characteristics and access patterns to identify the correct configuration for your cloud resources. Key characteristics to consider include:
  - **Data type:** structured, semi-structured, unstructured
  - **Data growth:** bounded, unbounded
  - **Data durability:** persistent, ephemeral, transient

  - **Access patterns** reads or writes, update frequency, spiky, or consistent
- **Use optimal architecture patterns:** Use architecture patterns that best support data access and storage patterns.
  - Patterns for enabling data persistence
  - Let's Architect! Modern data architectures
  - Databases on AWS: The Right Tool for the Right Job
- **Use purpose-built services:** Use technologies that are fit-for-purpose.
  - Use technologies that work natively with compressed data.
    - Athena Compression Support file formats
    - Format Options for ETL Inputs and Outputs in AWS Glue
    - Loading compressed data files from Amazon S3 with Amazon Redshift
  - Use purpose-built analytics services for data processing in your architecture. For detail on AWS purpose-built analytics services, see AWS re:Invent 2022 - Building modern data architectures on AWS.
  - Use the database engine that best supports your dominant query pattern. Manage your database indexes for efficient querying. For further details, see AWS Databases and AWS re:Invent 2022 - Modernize apps with purpose-built databases.
- **Minimize data transfer:** Select network protocols that reduce the amount of network capacity consumed in your architecture.

Resources

Related documents:

- COPY from columnar data formats with Amazon Redshift
- Converting Your Input Record Format in Firehose
- Improve query performance on Amazon Athena by Converting to Columnar Formats
- Monitoring DB load with Performance Insights on Amazon Aurora
- Monitoring DB load with Performance Insights on Amazon RDS
- Amazon S3 Intelligent-Tiering storage class
- Build a CQRS event store with Amazon DynamoDB

Related videos:

- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge

Related examples:

- AWS Purpose Built Databases Workshop
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS

Data

Question

- SUS 4 How do you take advantage of data management policies and patterns to support your sustainability goals?

SUS 4 How do you take advantage of data management policies and patterns to support your sustainability goals?

Implement data management practices to reduce the provisioned storage required to support your workload, and the resources required to use it. Understand your data, and use storage technologies and configurations that more effectively support the business value of the data and how it's used. Lifecycle data to more efficient, less performant storage when requirements decrease, and delete data that's no longer required.

Best practices

- SUS04-BP01 Implement a data classification policy
- SUS04-BP02 Use technologies that support data access and storage patterns
- SUS04-BP03 Use policies to manage the lifecycle of your datasets
- SUS04-BP04 Use elasticity and automation to expand block storage or file system

- SUS04-BP05 Remove unneeded or redundant data
- SUS04-BP06 Use shared file systems or storage to access common data
- SUS04-BP07 Minimize data movement across networks
- SUS04-BP08 Back up data only when difficult to recreate

SUS04-BP01 Implement a data classification policy

Classify data to understand its criticality to business outcomes and choose the right energy-efficient storage tier to store the data.

Common anti-patterns:

- You do not identify data assets with similar characteristics (such as sensitivity, business criticality, or regulatory requirements) that are being processed or stored.
- You have not implemented a data catalog to inventory your data assets.

Benefits of establishing this best practice: Implementing a data classification policy allows you to determine the most energy-efficient storage tier for data.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Data classification involves identifying the types of data that are being processed and stored in an information system owned or operated by an organization. It also involves making a determination on the criticality of the data and the likely impact of a data compromise, loss, or misuse.

Implement data classification policy by working backwards from the contextual use of the data and creating a categorization scheme that takes into account the level of criticality of a given dataset to an organization's operations.

Implementation steps

- **Perform data inventory:** Conduct an inventory of the various data types that exist for your workload.
- **Group data:** Determine criticality, confidentiality, integrity, and availability of data based on risk to the organization. Use these requirements to group data into one of the data classification

tiers that you adopt. As an example, see Four simple steps to classify your data and secure your startup.

- **Define data classification levels and policies:** For each data group, define data classification level (for example, public or confidential) and handling policies. Tag data accordingly. For more detail on data classification categories, see Data Classification whitepaper.
- **Periodically review:** Periodically review and audit your environment for untagged and unclassified data. Use automation to identify this data, and classify and tag the data appropriately. As an example, see Data Catalog and crawlers in AWS Glue.
- **Establish a data catalog:** Establish a data catalog that provides audit and governance capabilities.
- **Documentation:** Document data classification policies and handling procedures for each data class.

Resources

Related documents:

- Leveraging AWS Cloud to Support Data Classification
- Tag policies from AWS Organizations

Related videos:

- AWS re:Invent 2022 - Enabling agility with data governance on AWS
- AWS re:Invent 2023 - Data protection and resilience with AWS storage

SUS04-BP02 Use technologies that support data access and storage patterns

Use storage technologies that best support how your data is accessed and stored to minimize the resources provisioned while supporting your workload.

Common anti-patterns:

- You assume that all workloads have similar data storage and access patterns.
- You only use one tier of storage, assuming all workloads fit within that tier.
- You assume that data access patterns will stay consistent over time.

Benefits of establishing this best practice: Selecting and optimizing your storage technologies based on data access and storage patterns will help you reduce the required cloud resources to meet your business needs and improve the overall efficiency of cloud workload.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Select the storage solution that aligns best to your access patterns, or consider changing your access patterns to align with the storage solution to maximize performance efficiency.

Implementation steps

- **Evaluate data and access characteristics:** Evaluate your data characteristics and access pattern to collect the key characteristics of your storage needs. Key characteristics to consider include:
  - **Data type:** structured, semi-structured, unstructured
  - **Data growth:** bounded, unbounded
  - **Data durability:** persistent, ephemeral, transient
  - **Access patterns:** reads or writes, frequency, spiky, or consistent
- **Choose the right storage technology:** Migrate data to the appropriate storage technology that supports your data characteristics and access pattern. Here are some examples of AWS storage technologies and their key characteristics:

Type	Technology	Key characteristics
Object storage	Amazon S3	An object storage service with unlimited scalability, high availability, and multiple options for accessibility. Transferring and accessing objects in and out of Amazon S3 can use a service, such as Transfer Acceleration or Access Points, to support your location, security needs, and access patterns.

Type	Technology	Key characteristics
Archiving storage	Amazon Glacier	Storage class of Amazon S3 built for data-archiving.
Shared file system	Amazon Elastic File System (Amazon EFS)	Mountable file system that can be accessed by multiple types of compute solutions . Amazon EFS automatically grows and shrinks storage and is performance-optimi zed to deliver consistent low latencies.
Shared file system	Amazon FSx	Built on the latest AWS compute solutions to support four commonly used file systems: NetApp ONTAP, OpenZFS, Windows File Server, and Lustre. Amazon FSx latency, throughput, and IOPS vary per file system and should be considered when selecting the right file system for your workload needs.
Block storage	Amazon Elastic Block Store (Amazon EBS)	Scalable, high-performance block-storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). Amazon EBS includes SSD-backed storage for transactional, IOPS-intensive workloads and HDD-backe d storage for throughput-intensive workloads.

Type	Technology	Key characteristics
Relational database	Amazon Aurora, Amazon RDS, Amazon Redshift	Designed to support ACID (atomicity, consistency, isolation, durability) transactions and maintain referential integrity and strong data consistency. Many traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), and ecommerce systems use relational databases to store their data.
Key-value database	Amazon DynamoDB	Optimized for common access patterns, typically to store and retrieve large volumes of data. High-traffic web apps, ecommerce systems, and gaming applications are typical use-cases for key-value databases.

- **Automate storage allocation:** For storage systems that are a fixed size, such as Amazon EBS or Amazon FSx, monitor the available storage space and automate storage allocation on reaching a threshold. You can leverage Amazon CloudWatch to collect and analyze different metrics for Amazon EBS and Amazon FSx.
- **Choose the right storage class:** Choose the appropriate storage class for your data.
  - Amazon S3 storage classes can be configured at the object level. A single bucket can contain objects stored across all of the storage classes.
  - You can use Amazon S3 Lifecycle policies to automatically transition objects between storage classes or remove data without any application changes. In general, you have to make a trade-off between resource efficiency, access latency, and reliability when considering these storage mechanisms.

Resources

Related documents:

- Amazon EBS volume types
- Amazon EC2 instance store
- Amazon S3 Intelligent-Tiering
- Amazon EBS I/O Characteristics
- Using Amazon S3 storage classes
- What is Amazon Glacier?

Related videos:

- AWS re:Invent 2023 - Improve Amazon EBS efficiency and be more cost-efficient
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3
- AWS re:Invent 2022 - Building modern data architectures on AWS
- AWS re:Invent 2022 - Modernize apps with purpose-built databases
- AWS re:Invent 2022 - Building data mesh architectures on AWS
- AWS re:Invent 2023 - Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023 - Advanced data modeling with Amazon DynamoDB

Related examples:

- Amazon S3 Examples
- AWS Purpose Built Databases Workshop
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS

SUS04-BP03 Use policies to manage the lifecycle of your datasets

Manage the lifecycle of all of your data and automatically enforce deletion to minimize the total storage required for your workload.

Common anti-patterns:

- You manually delete data.
- You do not delete any of your workload data.
- You do not move data to more energy-efficient storage tiers based on its retention and access requirements.

Benefits of establishing this best practice: Using data lifecycle policies ensures efficient data access and retention in a workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Datasets usually have different retention and access requirements during their lifecycle. For example, your application may need frequent access to some datasets for a limited period of time. After that, those datasets are infrequently accessed. To improve the efficiency of data storage and computation over time, implement lifecycle policies, which are rules that define how data is handled over time.

With lifecycle configuration rules, you can tell the specific storage service to transition a dataset to more energy-efficient storage tiers, archive it, or delete it. This practice minimizes active data storage and retrieval, which leads to lower energy consumption. In addition, practices such as archiving or deleting obsolete data support regulatory compliance and data governance.

Implementation steps

- **Use data classification:** Classify datasets in your workload.
- **Define handling rules:** Define handling procedures for each data class.
- **Enable automation:** Set automated lifecycle policies to enforce lifecycle rules. Here are some examples of how to set up automated lifecycle policies for different AWS storage services:

Storage service	How to set automated lifecycle policies
Amazon S3	You can use Amazon S3 Lifecycle to manage your objects throughout their lifecycle. If your access patterns are unknown, changing, or unpredictable, you can use Amazon S3

Storage service	How to set automated lifecycle policies
	Intelligent-Tiering, which monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers. You can leverage Amazon S3 Storage Lens metrics to identify optimization opportunities and gaps in lifecycle management.
Amazon Elastic Block Store	You can use Amazon Data Lifecycle Manager to automate the creation, retention, and deletion of Amazon EBS snapshots and Amazon EBS-backed AMIs.
Amazon Elastic File System	Amazon EFS lifecycle management automatically manages file storage for your file systems.
Amazon Elastic Container Registry	Amazon ECR lifecycle policies automate the cleanup of your container images by expiring images based on age or count.
AWS Elemental MediaStore	You can use an object lifecycle policy that governs how long objects should be stored in the MediaStore container.

- **Delete unused assets:** Delete unused volumes, snapshots, and data that is out of its retention period. Use native service features like Amazon DynamoDB Time To Live or Amazon CloudWatch log retention for deletion.
- **Aggregate and compress:** Aggregate and compress data where applicable based on lifecycle rules.

Resources

Related documents:

- Optimize your Amazon S3 Lifecycle rules with Amazon S3 Storage Class Analysis
- Evaluating Resources with AWS Config Rules

Related videos:

- AWS re:Invent 2021 - Amazon S3 Lifecycle best practices to optimize your storage spend
- AWS re:Invent 2023 - Optimizing storage price and performance with Amazon S3
- Simplify Your Data Lifecycle and Optimize Storage Costs With Amazon S3 Lifecycle
- Reduce Your Storage Costs Using Amazon S3 Storage Lens

SUS04-BP04 Use elasticity and automation to expand block storage or file system

Use elasticity and automation to expand block storage or file system as data grows to minimize the total provisioned storage.

Common anti-patterns:

- You procure large block storage or file system for future need.
- You overprovision the input and output operations per second (IOPS) of your file system.
- You do not monitor the utilization of your data volumes.

Benefits of establishing this best practice: Minimizing over-provisioning for storage system reduces the idle resources and improves the overall efficiency of your workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Create block storage and file systems with size allocation, throughput, and latency that are appropriate for your workload. Use elasticity and automation to expand block storage or file system as data grows without having to over-provision these storage services.

Implementation steps

- For fixed size storage like Amazon EBS, verify that you are monitoring the amount of storage used versus the overall storage size and create automation, if possible, to increase the storage size when reaching a threshold.
- Use elastic volumes and managed block data services to automate allocation of additional storage as your persistent data grows. As an example, you can use Amazon EBS Elastic Volumes to change volume size, volume type, or adjust the performance of your Amazon EBS volumes.

- Choose the right storage class, performance mode, and throughput mode for your file system to address your business need, not exceeding that.
  - Amazon EFS performance
  - Amazon EBS volume performance on Linux instances
- Set target levels of utilization for your data volumes, and resize volumes outside of expected ranges.
- Right size read-only volumes to fit the data.
- Migrate data to object stores to avoid provisioning the excess capacity from fixed volume sizes on block storage.
- Regularly review elastic volumes and file systems to terminate idle volumes and shrink over-provisioned resources to fit the current data size.

Resources

Related documents:

- Extend the file system after resizing an EBS volume
- Modify a volume using Amazon EBS Elastic Volumes
- Amazon FSx Documentation
- What is Amazon Elastic File System?

Related videos:

- Deep Dive on Amazon EBS Elastic Volumes
- Amazon EBS and Snapshot Optimization Strategies for Better Performance and Cost Savings
- Optimizing Amazon EFS for cost and performance, using best practices

SUS04-BP05 Remove unneeded or redundant data

Remove unneeded or redundant data to minimize the storage resources required to store your datasets.

Common anti-patterns:

- You duplicate data that can be easily obtained or recreated.

- You back up all data without considering its criticality.
- You only delete data irregularly, on operational events, or not at all.
- You store data redundantly irrespective of the storage service's durability.
- You turn on Amazon S3 versioning without any business justification.

Benefits of establishing this best practice: Removing unneeded data reduces the storage size required for your workload and the workload environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

When you remove unneeded and redundant datasets, you can reduce storage cost and environmental footprint. This practice may also make computing more efficient, as compute resources only process important data instead of unneeded data. Automate the deletion of unneeded data. Use technologies that deduplicate data at the file and block level. Use service features for native data replication and redundancy.

Implementation steps

- **Evaluate public datasets:** Evaluate if you can avoid storing data by using existing publicly available datasets in AWS Data Exchange and Open Data on AWS.
- **De-deplicate data:** Use mechanisms that can deduplicate data at the block and object level. Here are some examples of how to deduplicate data on AWS:

Storage service	Deduplication mechanism
Amazon S3	Use AWS Lake Formation FindMatches to find matching records across a dataset (includin g ones without identifiers) by using the new FindMatches ML Transform.
Amazon FSx	Use data deduplication on Amazon FSx for Windows.
Amazon Elastic Block Store snapshots	Snapshots are incremental backups, which means that only the blocks on the device

Storage service	Deduplication mechanism
	that have changed after your most recent snapshot are saved.

- **Use lifecycle policies:** Use lifecycle policies to automate unneeded data deletion. Use native service features like Amazon DynamoDB Time To Live, Amazon S3 Lifecycle, or Amazon CloudWatch log retention for deletion.
- **Use data virtualization:** Use data virtualization capabilities on AWS to maintain data at its source and avoid data duplication.
  - Cloud Native Data Virtualization on AWS
  - Optimize Data Pattern Using Amazon Redshift Data Sharing
- **Use incremental backup:** Use backup technology that can make incremental backups.
- **Use native durability:** Leverage the durability of Amazon S3 and replication of Amazon EBS to meet your durability goals instead of self-managed technologies (such as a redundant array of independent disks (RAID)).
- **Use efficient logging:** Centralize log and trace data, deduplicate identical log entries, and establish mechanisms to tune verbosity when needed.
- **Use efficient caching:** Pre-populate caches only where justified.
- Establish cache monitoring and automation to resize the cache accordingly.
- **Remove old version assets:** Remove out-of-date deployments and assets from object stores and edge caches when pushing new versions of your workload.

Resources

Related documents:

- Change log data retention in CloudWatch Logs
- Data deduplication on Amazon FSx for Windows File Server
- Features of Amazon FSx for ONTAP including data deduplication
- Invalidating Files on Amazon CloudFront
- Using AWS Backup to back up and restore Amazon EFS file systems
- What is Amazon CloudWatch Logs?
- Working with backups on Amazon RDS

- Integrate and deduplicate datasets using AWS Lake Formation

Related videos:

- Amazon Redshift Data Sharing Use Cases

Related examples:

- How do I analyze my Amazon S3 server access logs using Amazon Athena?

SUS04-BP06 Use shared file systems or storage to access common data

Adopt shared file systems or storage to avoid data duplication and allow for more efficient infrastructure for your workload.

Common anti-patterns:

- You provision storage for each individual client.
- You do not detach data volume from inactive clients.
- You do not provide access to storage across platforms and systems.

Benefits of establishing this best practice: Using shared file systems or storage allows for sharing data to one or more consumers without having to copy the data. This helps to reduce the storage resources required for the workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

If you have multiple users or applications accessing the same datasets, using shared storage technology is crucial to use efficient infrastructure for your workload. Shared storage technology provides a central location to store and manage datasets and avoid data duplication. It also enforces consistency of the data across different systems. Moreover, shared storage technology allows for more efficient use of compute power, as multiple compute resources can access and process data at the same time in parallel.

Fetch data from these shared storage services only as needed and detach unused volumes to free up resources.

Implementation steps

- **Use shared storage:** Migrate data to shared storage when the data has multiple consumers. Here are some examples of shared storage technology on AWS:

Storage option	When to use
Amazon EBS Multi-Attach	Amazon EBS Multi-Attach allows you to attach a single Provisioned IOPS SSD (io1 or io2) volume to multiple instances that are in the same Availability Zone.
Amazon EFS	See When to Choose Amazon EFS.
Amazon FSx	See Choosing an Amazon FSx File System.
Amazon S3	Applications that do not require a file system structure and are designed to work with object storage can use Amazon S3 as a massively scalable, durable, low-cost object storage solution.

- **Fetch data as needed:** Copy data to or fetch data from shared file systems only as needed. As an example, you can create an Amazon FSx for Lustre file system backed by Amazon S3 and only load the subset of data required for processing jobs to Amazon FSx.
- **Delete unneeded data:** Delete data as appropriate for your usage patterns as outlined in SUS04-BP03 Use policies to manage the lifecycle of your datasets.
- **Detach inactive clients:** Detach volumes from clients that are not actively using them.

Resources

Related documents:

- Linking your file system to an Amazon S3 bucket
- Using Amazon EFS for AWS Lambda in your serverless applications
- Amazon EFS Intelligent-Tiering Optimizes Costs for Workloads with Changing Access Patterns
- Using Amazon FSx with your on-premises data repository

related videos:

- Storage cost optimization with Amazon EFS
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - File storage for builders and data scientists on Amazon Elastic File System

SUS04-BP07 Minimize data movement across networks

Use shared file systems or object storage to access common data and minimize the total networking resources required to support data movement for your workload.

Common anti-patterns:

- You store all data in the same AWS Region independent of where the data users are.
- You do not optimize data size and format before moving it over the network.

Benefits of establishing this best practice: Optimizing data movement across the network reduces the total networking resources required for the workload and lowers its environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Moving data around your organization requires compute, networking, and storage resources. Use techniques to minimize data movement and improve the overall efficiency of your workload.

Implementation steps

- **Use proximity:** Consider proximity to the data or users as a decision factor when selecting a Region for your workload.
- **Partition services:** Partition Regionally-consumed services so that their Region-specific data is stored within the Region where it is consumed.
- **Use efficient file formats:** Use efficient file formats (such as Parquet or ORC) and compress data before you move it over the network.
- **Minimize data movement:** Don't move unused data. Some examples that can help you avoid moving unused data:
  - Reduce API responses to only relevant data.

  - Aggregate data where detailed (record-level information is not required).
  - See Well-Architected Lab - Optimize Data Pattern Using Amazon Redshift Data Sharing.
  - Consider Cross-account data sharing in AWS Lake Formation.
- **Use edge services:** Use services that can help you run code closer to users of your workload.

Service	When to use
Lambda@Edge	Use for compute-heavy operations that are run when objects are not in the cache.
CloudFront Functions	Use for simple use cases such as HTTP(s) request/response manipulations that can be initiated by short-lived functions.
AWS IoT Greengrass	Run local compute, messaging, and data caching for connected devices.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part III: Networking
- AWS Global Infrastructure
- Amazon CloudFront Key Features including the CloudFront Global Edge Network
- Compressing HTTP requests in Amazon OpenSearch Service
- Intermediate data compression with Amazon EMR
- Loading compressed data files from Amazon S3 into Amazon Redshift
- Serving compressed files with Amazon CloudFront

Related videos:

- Demystifying data transfer on AWS

SUS04-BP08 Back up data only when difficult to recreate

Avoid backing up data that has no business value to minimize storage resources requirements for your workload.

Common anti-patterns:

- You do not have a backup strategy for your data.
- You back up data that can be easily recreated.

Benefits of establishing this best practice: Avoiding back-up of non-critical data reduces the required storage resources for the workload and lowers its environmental impact.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Avoiding the back up of unnecessary data can help lower cost and reduce the storage resources used by the workload. Only back up data that has business value or is needed to satisfy compliance requirements. Examine backup policies and exclude ephemeral storage that doesn't provide value in a recovery scenario.

Implementation steps

- **Classify data:** Implement data classification policy as outlined in SUS04-BP01 Implement a data classification policy.
- **Design a backup strategy:** Use the criticality of your data classification and design backup strategy based on your recovery time objective (RTO) and recovery point objective (RPO). Avoid backing up non-critical data.
  - Exclude data that can be easily recreated.
  - Exclude ephemeral data from your backups.
  - Exclude local copies of data, unless the time required to restore that data from a common location exceeds your service-level agreements (SLAs).
- **Use automated backup:** Use an automated solution or managed service to back up business-critical data.
  - AWS Backup is a fully-managed service that makes it easy to centralize and automate data protection across AWS services, in the cloud, and on premises. For hands-on guidance on how

to create automated backups using AWS Backup, see Well-Architected Labs - Testing Backup and Restore of Data.

- Automate backups and optimize backup costs for Amazon EFS using AWS Backup.

Resources

Related best practices:

- REL09-BP01 Identify and back up all data that needs to be backed up, or reproduce the data from sources
- REL09-BP03 Perform data backup automatically
- REL13-BP02 Use defined recovery strategies to meet the recovery objectives

Related documents:

- Using AWS Backup to back up and restore Amazon EFS file systems
- Amazon EBS snapshots
- Working with backups on Amazon Relational Database Service
- APN Partner: partners that can help with backup
- AWS Marketplace: products that can be used for backup
- Backing Up Amazon EFS
- Backing Up Amazon FSx for Windows File Server
- Backup and Restore for Amazon ElastiCache (Redis OSS)

Related videos:

- AWS re:Invent 2023 - Backup and disaster recovery strategies for increased resilience
- AWS re:Invent 2023 - What's new with AWS Backup
- AWS re:Invent 2021 - Backup, disaster recovery, and ransomware protection with AWS

Hardware and services

Question

- SUS 5 How do you select and use cloud hardware and services in your architecture to support your sustainability goals?

SUS 5 How do you select and use cloud hardware and services in your architecture to support your sustainability goals?

Look for opportunities to reduce workload sustainability impacts by making changes to your hardware management practices. Minimize the amount of hardware needed to provision and deploy, and select the most efficient hardware and services for your individual workload.

Best practices

- SUS05-BP01 Use the minimum amount of hardware to meet your needs
- SUS05-BP02 Use instance types with the least impact
- SUS05-BP03 Use managed services
- SUS05-BP04 Optimize your use of hardware-based compute accelerators

SUS05-BP01 Use the minimum amount of hardware to meet your needs

Use the minimum amount of hardware for your workload to efficiently meet your business needs.

Common anti-patterns:

- You do not monitor resource utilization.
- You have resources with a low utilization level in your architecture.
- You do not review the utilization of static hardware to determine if it should be resized.
- You do not set hardware utilization goals for your compute infrastructure based on business KPIs.

Benefits of establishing this best practice: Rightsizing your cloud resources helps to reduce a workload's environmental impact, save money, and maintain performance benchmarks.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Optimally select the total number of hardware required for your workload to improve its overall efficiency. The AWS Cloud provides the flexibility to expand or reduce the number of resources

dynamically through a variety of mechanisms, such as AWS Auto Scaling, and meet changes in demand. It also provides APIs and SDKs that allow resources to be modified with minimal effort. Use these capabilities to make frequent changes to your workload implementations. Additionally, use rightsizing guidelines from AWS tools to efficiently operate your cloud resource and meet your business needs.

Implementation steps

- **Choose the instances type:** Choose the right instances type to best fit your needs. To learn about how to choose Amazon Elastic Compute Cloud instances and use mechanisms such as attribute-based instance selection, see the following:
  - How do I choose the appropriate Amazon EC2 instance type for my workload?
  - Attribute-based instance type selection for Amazon EC2 Fleet.
  - Create an Auto Scaling group using attribute-based instance type selection.
- **Scale:** Use small increments to scale variable workloads.
- **Use multiple compute purchase options:** Balance instance flexibility, scalability, and cost savings with multiple compute purchase options.
  - Amazon EC2 On-Demand Instances are best suited for new, stateful, and spiky workloads which can't be instance type, location, or time flexible.
  - Amazon EC2 Spot Instances are a great way to supplement the other options for applications that are fault tolerant and flexible.
  - Leverage Compute Savings Plans for steady state workloads that allow flexibility if your needs (like AZ, Region, instance families, or instance types) change.
- **Use instance and Availability Zone diversity:** Maximize application availability and take advantage of excess capacity by diversifying your instances and Availability Zones.
- **Rightsize instances:** Use the rightsizing recommendations from AWS tools to make adjustments on your workload. For more information, see Optimizing your cost with Rightsizing Recommendations and Right Sizing: Provisioning Instances to Match Workloads
  - Use rightsizing recommendations in AWS Cost Explorer or AWS Compute Optimizer to identify rightsizing opportunities.
- **Negotiate service-level agreements (SLAs):** Negotiate SLAs that permit temporarily reducing capacity while automation deploys replacement resources.

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- Attirbute based Instance Type Selection for Auto Scaling for Amazon EC2 Fleet
- AWS Compute Optimizer Documentation
- Operating Lambda: Performance optimization
- Auto Scaling Documentation

Related videos:

- AWS re:Invent 2023 - What's new with Amazon EC2
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2022 - Optimizing Amazon Elastic Kubernetes Service for performance and cost on AWS
- AWS re:Invent 2023 - Sustainable compute: reducing costs and carbon emissions with AWS

SUS05-BP02 Use instance types with the least impact

Continually monitor and use new instance types to take advantage of energy efficiency improvements.

Common anti-patterns:

- You are only using one family of instances.
- You are only using x86 instances.
- You specify one instance type in your Amazon EC2 Auto Scaling configuration.
- You use AWS instances in a manner that they were not designed for (for example, you use compute-optimized instances for a memory-intensive workload).
- You do not evaluate new instance types regularly.
- You do not check recommendations from AWS rightsizing tools such as AWS Compute Optimizer.

Benefits of establishing this best practice: By using energy-efficient and right-sized instances, you are able to greatly reduce the environmental impact and cost of your workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Using efficient instances in cloud workload is crucial for lower resource usage and cost-effectiveness. Continually monitor the release of new instance types and take advantage of energy efficiency improvements, including those instance types designed to support specific workloads such as machine learning training and inference, and video transcoding.

Implementation steps

- **Learn and explore instance types:** Find instance types that can lower your workload's environmental impact.
  - Subscribe to What's New with AWS to stay up-to-date with the latest AWS technologies and instances.
  - Learn about different AWS instance types.
  - Learn about AWS Graviton-based instances which offer the best performance per watt of energy use in Amazon EC2 by watching re:Invent 2020 - Deep dive on AWS Graviton2 processor-powered Amazon EC2 instances and Deep dive into AWS Graviton3 and Amazon EC2 C7g instances.
- **Use instance types with the least impact:** Plan and transition your workload to instance types with the least impact.
  - Define a process to evaluate new features or instances for your workload. Take advantage of agility in the cloud to quickly test how new instance types can improve your workload environmental sustainability. Use proxy metrics to measure how many resources it takes you to complete a unit of work.
  - If possible, modify your workload to work with different numbers of vCPUs and different amounts of memory to maximize your choice of instance type.
  - Consider transitioning your workload to Graviton-based instances to improve the performance efficiency of your workload. For more information on moving workloads to AWS Graviton, see AWS Graviton Fast Start and Considerations when transitioning workloads to AWS Graviton-based Amazon Elastic Compute Cloud instances.
  - Consider selecting the AWS Graviton option in your usage of AWS managed services.
  - Migrate your workload to Regions that offer instances with the least sustainability impact and still meet your business requirements.

  - For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1. AWS Inferentia instances such as Inf2 instances offer up to 50% better performance per watt over comparable Amazon EC2 instances.
  - Use Amazon SageMaker AI Inference Recommender to right size ML inference endpoint.
  - For spiky workloads (workloads with infrequent requirements for additional capacity), use burstable performance instances.
  - For stateless and fault-tolerant workloads, use Amazon EC2 Spot Instances to increase overall utilization of the cloud, and reduce the sustainability impact of unused resources.
- **Operate and optimize:** Operate and optimize your workload instance.
  - For ephemeral workloads, evaluate instance Amazon CloudWatch metrics such as `CPUUtilization` to identify if the instance is idle or under-utilized.
  - For stable workloads, check AWS rightsizing tools such as AWS Compute Optimizer at regular intervals to identify opportunities to optimize and right-size the instances. For further examples and recommendations, see the following labs:
    - Well-Architected Lab - Rightsizing Recommendations
    - Well-Architected Lab - Rightsizing with Compute Optimizer
    - Well-Architected Lab - Optimize Hardware Patterns and Observice Sustainability KPIs

Resources

Related documents:

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute
- AWS Graviton
- Amazon EC2 DL1
- Amazon EC2 Capacity Reservation Fleets
- Amazon EC2 Spot Fleet
- Functions: Lambda Function Configuration
- Attribute-based instance type selection for Amazon EC2 Fleet
- Building Sustainable, Efficient, and Cost-Optimized Applications on AWS
- How the Contino Sustainability Dashboard Helps Customers Optimize Their Carbon Footprint

Related videos:

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 = What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Deep dive into AWS Graviton3 and Amazon EC2 C7g instances
- AWS re:Invent 2022 - Build a cost-, energy-, and resource-efficient compute environment

Related examples:

- Solution: Guidance for Optimizing Deep Learning Workloads for Sustainability on AWS

SUS05-BP03 Use managed services

Use managed services to operate more efficiently in the cloud.

Common anti-patterns:

- You use Amazon EC2 instances with low utilization to run your applications.
- Your in-house team only manages the workload, without time to focus on innovation or simplifications.
- You deploy and maintain technologies for tasks that can run more efficiently on managed services.

Benefits of establishing this best practice:

- Using managed services shifts the responsibility to AWS, which has insights across millions of customers that can help drive new innovations and efficiencies.
- Managed service distributes the environmental impact of the service across many users because of the multi-tenet control planes.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Managed services shift responsibility to AWS for maintaining high utilization and sustainability optimization of the deployed hardware. Managed services also remove the operational and administrative burden of maintaining a service, which allows your team to have more time and focus on innovation.

Review your workload to identify the components that can be replaced by AWS managed services. For example, Amazon RDS, Amazon Redshift, and Amazon ElastiCache provide a managed database service. Amazon Athena, Amazon EMR, and Amazon OpenSearch Service provide a managed analytics service.

Implementation steps

1. **Inventory your workload:** Inventory your workload for services and components.
2. **Identify candidates:** Assess and identify components that can be replaced by managed services. Here are some examples of when you might consider using a managed service:

Task	What to use on AWS
Hosting a database	Use managed Amazon Relational Database Service (Amazon RDS) instances instead of maintaining your own Amazon RDS instances on Amazon Elastic Compute Cloud (Amazon EC2).
Hosting a container workload	Use AWS Fargate, instead of implementing your own container infrastructure.
Hosting web apps	Use AWS Amplify Hosting as fully managed CI/CD and hosting service for static websites and server-side rendered web apps.

3. **Create a migration plan:** Identify dependencies and create a migrations plan. Update runbooks and playbooks accordingly.
    - The AWS Application Discovery Service automatically collects and presents detailed information about application dependencies and utilization to help you make more informed decisions as you plan your migration
4. **Perform tests** Test the service before migrating to the managed service.

5. **Replace self-hosted services:** Use your migration plan to replace self-hosted services with managed service.
6. **Monitor and adjust:** Continually monitor the service after the migration is complete to make adjustments as required and optimize the service.

Resources

Related documents:

- AWS Cloud Products
- AWS Total Cost of Ownership (TCO) Calculator
- Amazon DocumentDB
- Amazon Elastic Kubernetes Service (EKS)
- Amazon Managed Streaming for Apache Kafka (Amazon MSK)

Related videos:

- AWS re:Invent 2021 - Cloud operations at scale with AWS Managed Services
- AWS re:Invent 2023 - Best practices for operating on AWS

SUS05-BP04 Optimize your use of hardware-based compute accelerators

Optimize your use of accelerated computing instances to reduce the physical infrastructure demands of your workload.

Common anti-patterns:

- You are not monitoring GPU usage.
- You are using a general-purpose instance for workload while a purpose-built instance can deliver higher performance, lower cost, and better performance per watt.
- You are using hardware-based compute accelerators for tasks where they’re more efficient using CPU-based alternatives.

Benefits of establishing this best practice: By optimizing the use of hardware-based accelerators, you can reduce the physical-infrastructure demands of your workload.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

If you require high processing capability, you can benefit from using accelerated computing instances, which provide access to hardware-based compute accelerators such as graphics processing units (GPUs) and field programmable gate arrays (FPGAs). These hardware accelerators perform certain functions like graphic processing or data pattern matching more efficiently than CPU-based alternatives. Many accelerated workloads, such as rendering, transcoding, and machine learning, are highly variable in terms of resource usage. Only run this hardware for the time needed, and decommission them with automation when not required to minimize resources consumed.

Implementation steps

- **Explore compute accelerators:** Identify which accelerated computing instances can address your requirements.
- **Use purpose-built hardware:** For machine learning workloads, take advantage of purpose-built hardware that is specific to your workload, such as AWS Trainium, AWS Inferentia, and Amazon EC2 DL1. AWS Inferentia instances such as Inf2 instances offer up to 50% better performance per watt over comparable Amazon EC2 instances.
- **Monitor usage metrics:** Collect usage metric for your accelerated computing instances. For example, you can use CloudWatch agent to collect metrics such as `utilization_gpu` and `utilization_memory` for your GPUs as shown in Collect NVIDIA GPU metrics with Amazon CloudWatch.
- **Rightsize:** Optimize the code, network operation, and settings of hardware accelerators to make sure that underlying hardware is fully utilized.
  - Optimize GPU settings
  - GPU Monitoring and Optimization in the Deep Learning AMI
  - Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker AI
- **Keep up to date:** Use the latest high performant libraries and GPU drivers.
- **Release unneeded instances:** Use automation to release GPU instances when not in use.

Resources

Related documents:

- Accelerated Computing
- Let's Architect! Architecting with custom chips and accelerators
- How do I choose the appropriate Amazon EC2 instance type for my workload?
- Amazon EC2 VT1 Instances
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker AI

Related videos:

- AWS re:Invent 2021 - How to select Amazon EC2 GPU instances for deep learning
- AWS Online Tech Talks - Deploying Cost-Effective Deep Learning Inference
- AWS re:Invent 2023 - Cutting-edge AI with AWS and NVIDIA
- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances
- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

Process and culture

Question

- SUS 6 How do your organizational processes support your sustainability goals?

SUS 6 How do your organizational processes support your sustainability goals?

Look for opportunities to reduce your sustainability impact by making changes to your development, test, and deployment practices.

Best practices

- SUS06-BP01 Communicate and cascade your sustainability goals
- SUS06-BP02 Adopt methods that can rapidly introduce sustainability improvements
- SUS06-BP03 Keep your workload up-to-date
- SUS06-BP04 Increase utilization of build environments

- SUS06-BP05 Use managed device farms for testing

SUS06-BP01 Communicate and cascade your sustainability goals

Technology is a key enabler of sustainability. IT teams play a crucial role in driving meaningful change towards your organization's sustainability goals. These teams should clearly understand the company's sustainability targets and work to communicate and cascade those priorities across its operations.

Common anti-patterns:

- You don't know your organization's sustainability goals and how they apply to your team.
- You have insufficient awareness and training about the environmental impact of cloud workloads.
- You are unsure about the specific areas to prioritize.
- You do not involve your employees and customers in your sustainability initiatives.

Benefits of establishing this best practice: From optimization of infrastructure and systems to use of innovative technologies, IT teams can reduce the organization's carbon emissions and minimize resource consumption. Communication of sustainability goals can provide the ability for IT teams continuously improve and adapt to evolving sustainability challenges. Additionally, these sustainable optimizations often translate to cost savings as well, which strengthens the business case.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

The primary sustainability goals for IT teams should be to optimize systems and solutions to increase resource efficiency and minimize the organization's carbon footprint and overall environmental impact. Shared services and initiatives like training programs and operational dashboards can support organizations as they optimize IT operations and build solutions that can help significantly reduce the carbon footprint. The cloud presents an opportunity not only to move physical infrastructure and energy procurement responsibilities to the shared responsibility of the cloud provider but also to continuously optimize the resource efficiency of cloud-based services.

When teams use the cloud's inherent efficiency and shared responsibility model, they can drive meaningful reductions in the organization's environmental impact. This, in turn, can contribute

to the organization's overall sustainability goals and demonstrate the value of these teams as strategic partners in the journey towards a more sustainable future.

Implementation steps

- **Define goals and objectives:** Establish well-defined goals for your IT program. This involves getting input from responsible stakeholders from different departments such as IT, sustainability, and finance. These teams should define measurable goals that align with your organization's sustainability goals, including areas such carbon reduction and resource optimization.
- **Understand the carbon accounting boundaries of your business:** Understand how carbon accounting methods like the Greenhouse Gas (GHG) Protocol relate to your workloads in the cloud (for more detail, see Cloud sustainability).
- **Use cloud solutions for carbon accounting:** Use cloud solutions such as carbon accounting solutions on AWS to track scope one, two, and three for GHG emissions across your operations, portfolios, and value chains. With these solutions, organizations can streamline GHG emission data acquisition, simplify reporting, and derive insights to inform their climate strategies.
- **Monitor the carbon footprint of your IT portfolio:** Track and report carbon emissions of your IT systems. Use the AWS Customer Carbon Footprint Tool to track, measure, review, and forecast the carbon emissions generated from your AWS usage.
- **Communicate resource usage through proxy metrics to your teams:** Track and report on your resource usage through proxy metrics. In the on-demand pricing models of the cloud, resource usage is related to cost, which is a generally-understandable metric. At a minimum, use cost as a proxy metric to communicate the resource usage and improvements by each team.
  - **Enable hourly granularity in your Cost Explorer and create a Cost and Usage Report (CUR):** The CUR provides daily or hourly usage granularity, rates, costs, and usage attributes for all AWS services. Use the Cloud Intelligence Dashboards and its Sustainability Proxy Metrics Dashboard as a starting point for the processing and visualization of cost and usage based data. For more detail, see the following:
  - Measure and track cloud efficiency with sustainability proxy metrics, Part I: What are proxy metrics?
  - Measure and track cloud efficiency with sustainability proxy metrics, Part II: Establish a metrics pipeline
- **Continuously optimize and evaluate:** Use an improvement process to continuously optimize your IT systems, including cloud workload for efficiency and sustainability. Monitor carbon footprint before and after implementation of optimization strategy. Use the reduction in carbon footprint to assess the effectiveness.

- **Foster a sustainability culture:** Use training programs (like AWS Skill Builder) to educate your employees about sustainability. Engage them in sustainability initiatives. Share and celebrate their success stories. Use incentives to award them if they achieve sustainability targets.

Resources

Related documents:

- Understanding your carbon emission estimations

Related videos:

- AWS re:Invent 2023 - Accelerate data-driven circular economy initiatives with AWS
- AWS re:Invent 2023 - Sustainability innovation in AWS Global Infrastructure
- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure

Related examples:

- Well-Architected Lab - Turning cost & usage reports into efficiency reports

Related trainings:

- Sustainability Transformation on AWS
- SimuLearn - Sustainability Reporting
- Decarbonization with AWS

SUS06-BP02 Adopt methods that can rapidly introduce sustainability improvements

Adopt methods and processes to validate potential improvements, minimize testing costs, and deliver small improvements.

Common anti-patterns:

- Reviewing your application for sustainability is a task done only once at the beginning of a project.
- Your workload has become stale, as the release process is too cumbersome to introduce minor changes for resource efficiency.
- You do not have mechanisms to improve your workload for sustainability.

Benefits of establishing this best practice: By establishing a process to introduce and track sustainability improvements, you will be able to continually adopt new features and capabilities, remove issues, and improve workload efficiency.

Level of risk exposed if this best practice is not established: Medium

Implementation guidance

Test and validate potential sustainability improvements before deploying them to production. Account for the cost of testing when calculating potential future benefit of an improvement. Develop low cost testing methods to deliver small improvements.

Implementation steps

- **Understand and communicate your organizational sustainability goals:** Understand your organizational sustainability goals, such carbon reduction or water stewardship. Translate these goals into sustainability requirements for your cloud workloads. Communicate these requirements to key stakeholders.
- **Add sustainability requirements to your backlog:** Add requirements for sustainability improvement to your development backlog.
- **Iterate and improve:** Use an iterative improvement process to identify, evaluate, prioritize, test, and deploy these improvements.
- **Test using minimum viable product (MVP):** Develop and test potential improvements using the minimum viable representative components to reduce the cost and environmental impact of testing.
- **Streamline the process:** Continually improve and streamline your development processes. As an example, Automate your software delivery process using continuous integration and delivery (CI/CD) pipelines to test and deploy potential improvements to reduce the level of effort and limit errors caused by manual processes.
- **Training and awareness:** Run training programs for your team members to educate them about sustainability and how their activities impact your organizational sustainability goals.

- **Assess and adjust:** Continually assess the impact of improvements and make adjustments as needed.

Resources

Related documents:

- AWS enables sustainability solutions

Related videos:

- AWS re:Invent 2023 - Sustainable architecture: Past, present, and future
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2022 - Architecting sustainably and reducing your AWS carbon footprint
- AWS re:Invent 2022 - Sustainability in AWS global infrastructure
- AWS re:Invent 2023 - What's new with AWS observability and operations

SUS06-BP03 Keep your workload up-to-date

Keep your workload up-to-date to adopt efficient features, remove issues, and improve the overall efficiency of your workload.

Common anti-patterns:

- You assume your current architecture is static and will not be updated over time.
- You do not have any systems or a regular cadence to evaluate if updated software and packages are compatible with your workload.

Benefits of establishing this best practice: By establishing a process to keep your workload up to date, you can adopt new features and capabilities, resolve issues, and improve workload efficiency.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Up to date operating systems, runtimes, middlewares, libraries, and applications can improve workload efficiency and make it easier to adopt more efficient technologies. Up to date software

might also include features to measure the sustainability impact of your workload more accurately, as vendors deliver features to meet their own sustainability goals. Adopt a regular cadence to keep your workload up to date with the latest features and releases.

Implementation steps

- **Define a process:** Use a process and schedule to evaluate new features or instances for your workload. Take advantage of agility in the cloud to quickly test how new features can improve your workload to:
  - Reduce sustainability impacts.
  - Gain performance efficiencies.
  - Remove barriers for a planned improvement.
  - Improve your ability to measure and manage sustainability impacts.
- **Conduct an inventory:** Inventory your workload software and architecture and identify components that need to be updated.
  - You can use AWS Systems Manager Inventory to collect operating system (OS), application, and instance metadata from your Amazon EC2 instances and quickly understand which instances are running the software and configurations required by your software policy and which instances need to be updated.
- **Learn the update procedure:** Understand how to update the components of your workload.

Workload component	How to update
Machine images	Use EC2 Image Builder to manage updates to Amazon Machine Images (AMIs) for Linux or Windows server images.
Container images	Use Amazon Elastic Container Registry (Amazon ECR) with your existing pipeline to manage Amazon Elastic Container Service (Amazon ECS) images.
AWS Lambda	AWS Lambda includes version management features.

- **Use automation:** Automate updates to reduce the level of effort to deploy new features and limit errors caused by manual processes.
  - You can use CI/CD to automatically update AMIs, container images, and other artifacts related to your cloud application.
  - You can use tools such as AWS Systems Manager Patch Manager to automate the process of system updates, and schedule the activity using AWS Systems Manager Maintenance Windows.

Resources

Related documents:

- AWS Architecture Center
- What's New with AWS
- AWS Developer Tools

Related videos:

- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance
- All Things Patch: AWS Systems Manager

SUS06-BP04 Increase utilization of build environments

Increase the utilization of resources to develop, test, and build your workloads.

Common anti-patterns:

- You manually provision or terminate your build environments.
- You keep your build environments running independent of test, build, or release activities (for example, running an environment outside of the working hours of your development team members).
- You over-provision resources for your build environments.

Benefits of establishing this best practice: By increasing the utilization of build environments, you can improve the overall efficiency of your cloud workload while allocating the resources to builders to develop, test, and build efficiently.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Use automation and infrastructure-as-code to bring build environments up when needed and take them down when not used. A common pattern is to schedule periods of availability that coincide with the working hours of your development team members. Your test environments should closely resemble the production configuration. However, look for opportunities to use instance types with burst capacity, Amazon EC2 Spot Instances, automatic scaling database services, containers, and serverless technologies to align development and test capacity with use. Limit data volume to just meet the test requirements. If using production data in test, explore possibilities of sharing data from production and not moving data across.

Implementation steps

- **Use infrastructure as code:** Use infrastructure as code to provision your build environments.
- **Use automation:** Use automation to manage the lifecycle of your development and test environments and maximize the efficiency of your build resources.
- **Maximize utilization**: Use strategies to maximize the utilization of development and test environments.
  - Use minimum viable representative environments to develop and test potential improvements.
  - Use serverless technologies if possible.
  - Use On-Demand Instances to supplement your developer devices.
  - Use instance types with burst capacity, Spot Instances, and other technologies to align build capacity with use.
  - Adopt native cloud services for secure instance shell access rather than deploying fleets of bastion hosts.
  - Automatically scale your build resources depending on your build jobs.

Resources

Related documents:

- AWS Systems Manager Session Manager
- Amazon EC2 Burstable performance instances
- What is AWS CloudFormation?
- What is AWS CodeBuild?
- Instance Scheduler on AWS

Related videos:

- AWS re:Invent 2023 - Continuous integration and delivery for AWS

SUS06-BP05 Use managed device farms for testing

Use managed device farms to efficiently test a new feature on a representative set of hardware.

Common anti-patterns:

- You manually test and deploy your application on individual physical devices.
- You do not use app testing service to test and interact with your apps (for example, Android, iOS, and web apps) on real, physical devices.

Benefits of establishing this best practice: Using managed device farms for testing cloud-enabled applications provides a number of benefits:

- They include more efficient features to test application on wide range of devices.
- They eliminate the need for in-house infrastructure for testing.
- They offer diverse device types, including older and less popular hardware, which eliminates the need for unnecessary device upgrades.

Level of risk exposed if this best practice is not established: Low

Implementation guidance

Using Managed device farms can help you to streamline the testing process for new features on a representative set of hardware. Managed device farms offer diverse device types including older, less popular hardware, and avoid customer sustainability impact from unnecessary device upgrades.

Implementation steps

- **Define testing requirements**: Define your testing requirements and plan (like test type, operating systems, and test schedule).
  - You can use Amazon CloudWatch RUM to collect and analyze client-side data and shape your testing plan.

- **Select a managed device farm:** Select a managed device farm that can support your testing requirements. For example, you can use AWS Device Farm to test and understand the impact of your changes on a representative set of hardware.
- **Use automation:** Use automation and continuous integration/continuous deployment (CI/CD) to schedule and run your tests.
  - Integrating AWS Device Farm with your CI/CD pipeline to run cross-browser Selenium tests
  - Building and testing iOS and iPadOS apps with AWS DevOps and mobile services
- **Review and adjust:** Continually review your testing results and make necessary improvements.

Resources

Related documents:

- AWS Device Farm device list
- Viewing the CloudWatch RUM dashboard

Related videos:

- AWS re:Invent 2023 - Improve your mobile and web app quality using AWS Device Farm
- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM

Related examples:

- AWS Device Farm Sample App for Android
- AWS Device Farm Sample App for iOS
- Appium Web tests for AWS Device Farm

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Copyright © 2024 Amazon Web Services, Inc. or its affiliates.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.