Fastly Help Guides

§ Adding and deleting user accounts (/guides/user-access-and-control/adding-and-deleting-user-accounts)

Fastly allows you to add users to your account, assigning them different roles and permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions) as appropriate. You can delete user accounts when you no longer want them to have access.

IMPORTANT: You must be assigned the role of superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions) to add users to or delete users from an account.

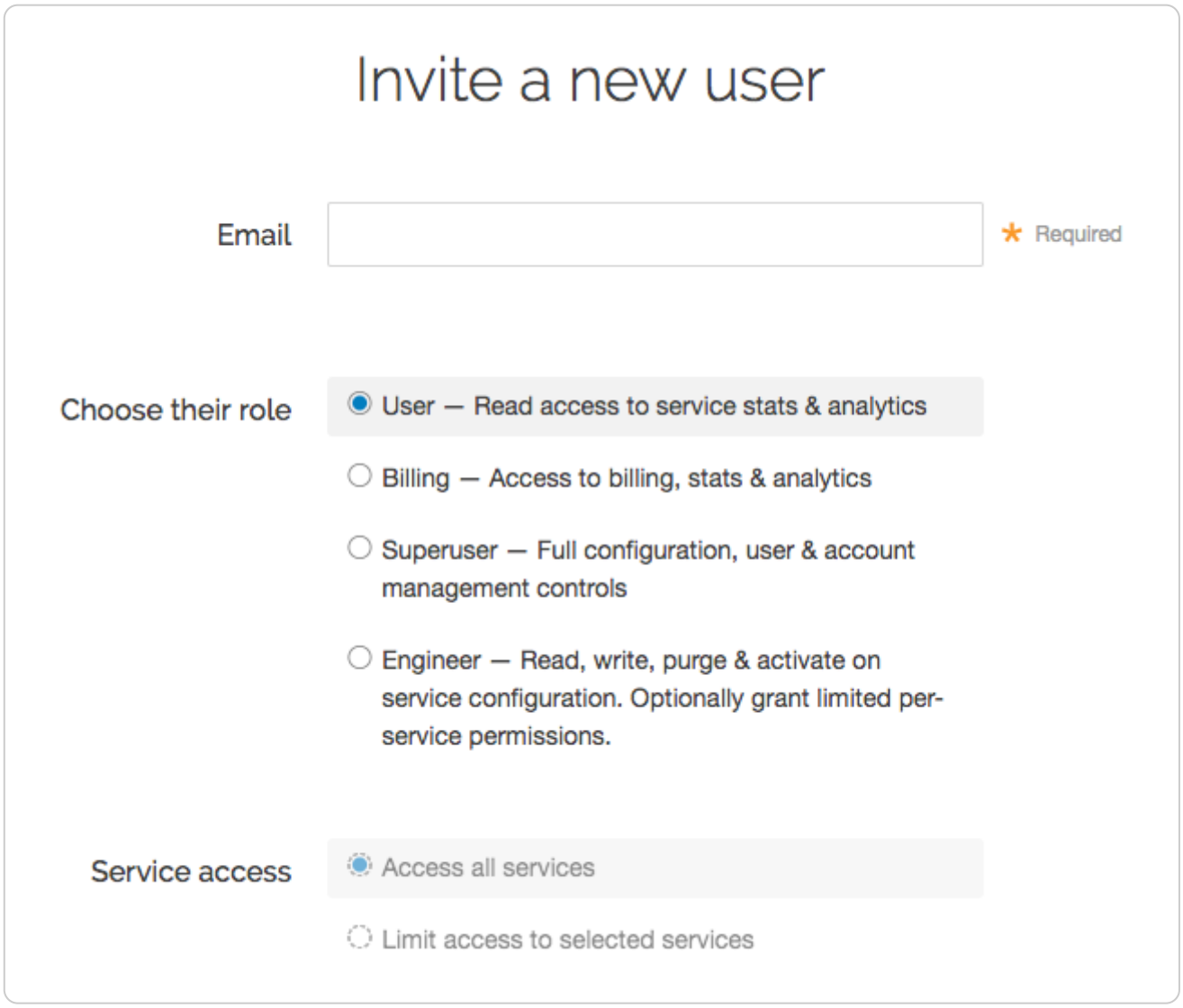
Adding account users

TIP: Adding a new user to make them the billing contact for your account? Follow our billing contact instructions (/guides/account-types-and-billing/who-receives-your-bill#changing-who-receives-your-bill) instead.

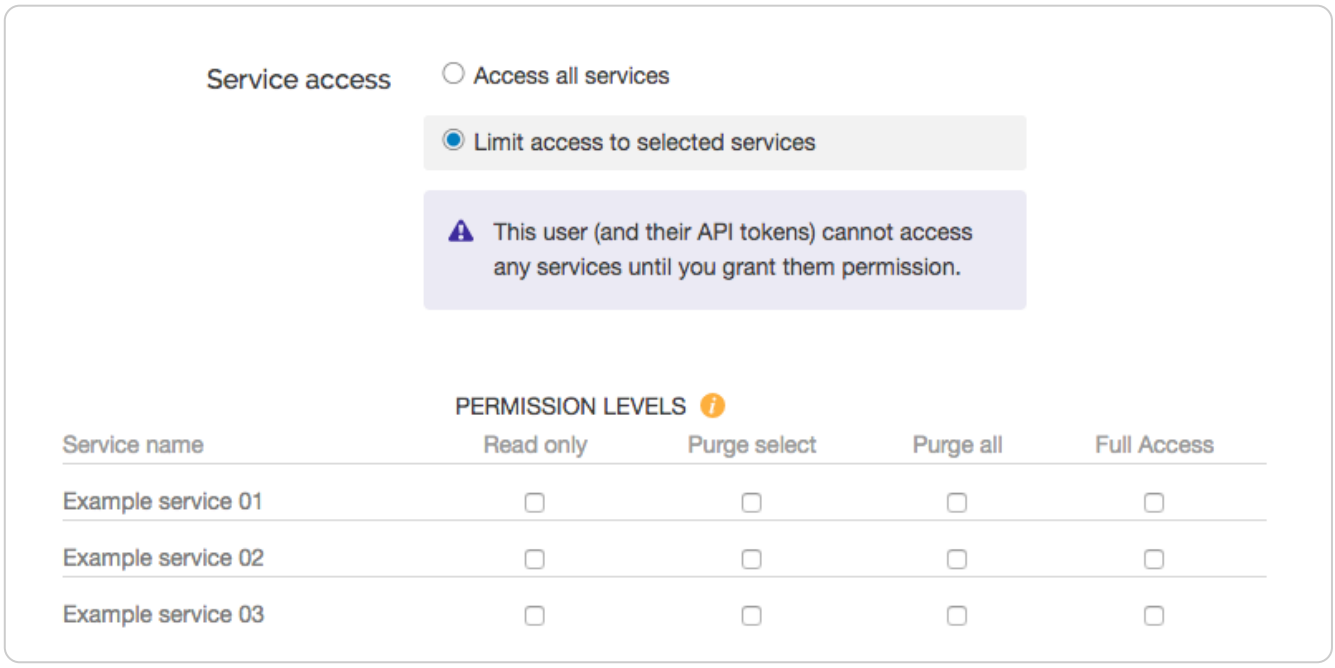
Adding a new user to your account

To add a new user to your account, send them an invitation to join following the steps below:

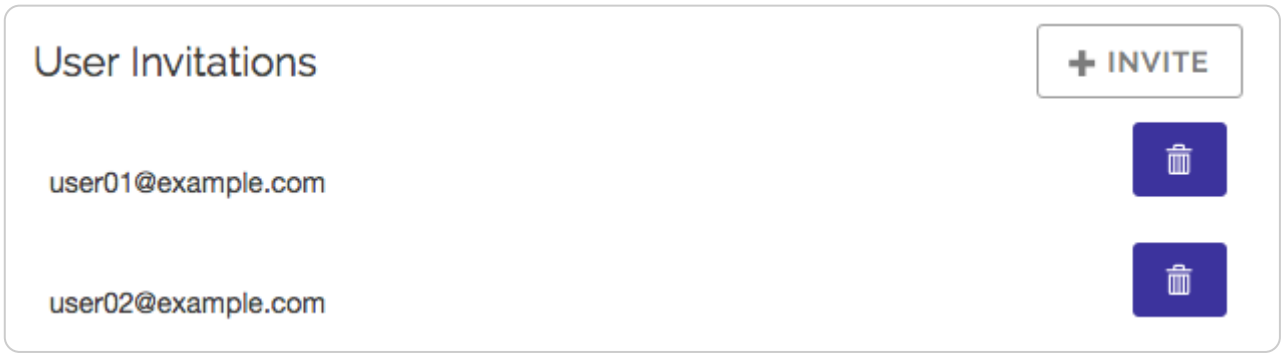
1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **User management** link.
3. In the **User Invitations** area, click the **Invite** button. The Invite a new user page appears.



4. In the **Email** field, type the email address of the user to invite.
5. From the **Choose their role** options, select the role to assign the user (/guides/user-access-and-control/configuring-user-roles-and-permissions#user-roles-and-what-they-can-do) once they accept the invitation.
6. From the **Service access** controls, optionally select **Limit access to selected services** to limit access to selected services (/guides/user-access-and-control/configuring-user-roles-and-permissions#service-access-and-permission-levels) for users assigned the role of engineer.



7. If you've chosen to limit access to selected services for a user assigned the role of engineer, select the specific permission levels (/guides/user-access-and-control/configuring-user-roles-and-permissions#service-access-and-permission-levels) for each service associated with the account.
8. Click the **Invite** button to send an invitation to the email you specified. The email address of the user you invited appears in the User Invitations area and remains there until the invitation is accepted.



Adding an existing user to your account

To add an existing user to your Fastly account, have them cancel their existing account (/guides/account-types-and-billing/accounts-and-pricing-plans#canceling-your-account) and then re-invite them by following the steps to add a new account user to your account. We associate a user's email address with an account. Canceling that account allows the email address to be reused.

NOTE: Account cancelation might not be an option in some situations. Contact support@fastly.com (mailto:support@fastly.com) to discuss how accounts can be combined.

Deleting account users

★ **TIP:** Deleting the owner of the account? Be sure to transfer ownership (/guides/user-access-and-control/configuring-user-roles-and-permissions#account-ownership-and-how-to-transfer-it) first.

To delete a user from your account, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **User management** link.
3. In the **Users** area, find the name of the user to delete.
4. Click the gear icon to the right of the user to be deleted, then select **Delete** from the menu that appears. A confirmation window appears.
5. If the user has active API tokens associated with their account, click the **Review this user's API tokens** link to manually review and revoke them. Alternatively, select the checkbox to automatically revoke all of the user's API tokens and delete the user.

   ⚠ **WARNING:** Deleting an API token will break any integration actively using that credential. Verify you have changed the API token for your integrations before proceeding.
6. Click **Confirm and delete**.

§ Configuring user roles and permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions)

Your Fastly account can be managed by multiple users. You can control each user's role, as well as control the scope of their service access and their specific permission levels for that service access.

★ **TIP:** The roles, service access, and permission levels you assign to users do not affect their ability to submit requests to Fastly Customer Support (/guides/detailed-product-descriptions/about-fastlys-customer-support).

User roles and what they can do

Fastly allows you to assign one of four different roles to each user allowed access to your account. In general, the abilities granted to each role are as follows:

- **User.** View stats and analytics information for all services on an account.
- **Billing.** View billing information about an account. View stats and analytics information for all services on an account.
- **Engineer.** View configuration details, issue purge requests, and make configuration changes, including activating new service versions. Some of these abilities may be restricted on a per service basis.
- **Superuser.** Full account access, including service configuration, user access and control, and account management capabilities for an account. Superusers cannot close or cancel an account unless they are also the account owner.

Abilities granted to user roles are selective, not additive. Specifically, each role has full (☑) or potentially restricted (⊟) access to the following functionality:

	User	Billing	Engineer	Superuser
Analytics & Stats				
View historical stats	X	X	X	X
View real-time service stats	X	X	X	X
Configure				
Create services			X	X
Delete services			?	X
Configure services			?	X
Compare service versions			?	X
Deactivate services			?	X
Purge			?	X
View and download generated VCL			?	X
Customize VCL			?	X

Account & Organization				
Update personal profile settings	X	X	X	X
Update company settings				X
Invite all new user roles				X
Invite new engineer and user roles (API only)			X	
Assign and change roles and permissions				X
Issue password resets				X
Delete account users				X
Enable and disable personal 2FA	X	X	X	X
Enable and disable company-wide 2FA				X
Manage personal API tokens	X	X	X	X
Revoke account API tokens				X
Billing				
View invoices		X		X
View billing history		X		X
Pay bills		X		X
Update credit card info		X		X
Change account type		X		X

Service access and permission levels

All user roles grant access by default to every service on an account now and in the future. The engineer role is unique, however, in that you can change that default. Superusers can limit an engineer's access to specific services and can control the level of permissions on each of those services as follows:

- **Read-only.** Allows an engineer to view a specific service's configuration but does not allow them to issue purge requests for that service nor make changes to its configuration.
- **Purge select.** Allows an engineer to view a specific service's configuration and also allows them to issue purge requests for that service via URL (/guides/purging/single-purges#purging-a-url) or surrogate key (/guides/purging/single-purges#purging-with-keys). They cannot use the purge all (/guides/purging/single-purges#purging-all-content) function on the service, nor can they make configuration changes to that service.

- **Purge all.** Allows an engineer to view a specific service's configuration and issue purge requests via URL, surrogate key, or the purge all function. They cannot, however, make configuration changes to that service.
- **Full access.** Allows an engineer full access to a specific service, including permission to issue purge requests via any method on that service. They can make configuration changes to that service and can activate new versions of it at will.

Permission levels are additive. Each level includes the previous level's permissions. When new services are added to an account by a superuser, engineers with limited access to services will not be granted permissions to those services until a superuser specifically grants those permission levels manually.

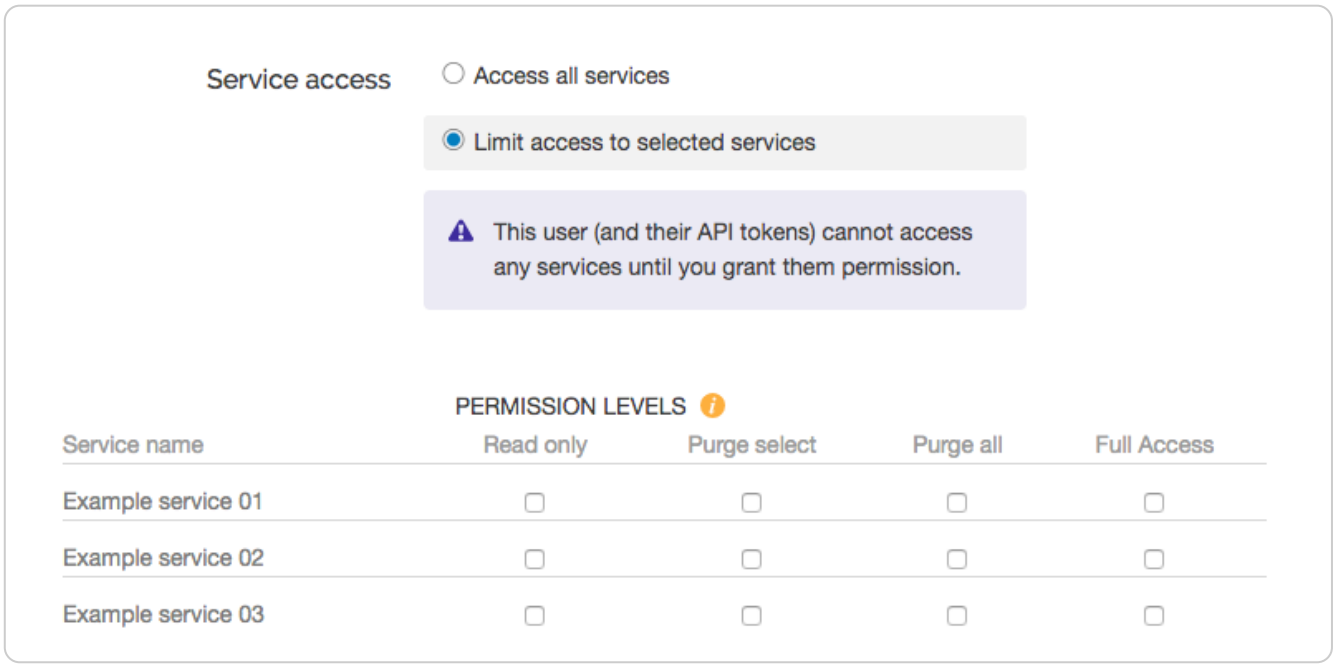
Users assigned the role of `engineer` can create new services (this is especially useful for learning about configuration options without affecting production services). By default, an engineer will automatically have full access to any service they create until their permission levels on that service are modified by an account superuser.

Changing user roles and access permissions for existing users

Users assigned the superuser role can change the role, service access, or permission levels for any existing user on your account. Plan your changes carefully.

> ⚠ **WARNING:** Role, service access, and permission level changes for existing users apply instantly and get saved automatically.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **User management** link. The User management page appears.
3. In the **Users** area, click the gear icon next to a user name and then select **Access controls** from the menu that appears. The Edit access control page appears for the selected user.
4. From the **Choose their role** choices, optionally select a new role for the user.
5. From the **Service access** controls, optionally select **Limit access to selected services** to limit access to selected services for users assigned the role of engineer.



6. If you've limited access to selected services for a user assigned the role of engineer, select the specific permission levels for each service associated with the account.
7. Click **Update**. The user's role and permission levels will be changed accordingly.

Account ownership and how to transfer it

We assign account "ownership" to the first user who signs up for an account for your organization. We automatically assign owners the superuser role, though that role can be changed by another superuser once additional users are added.

Accounts can only be canceled (/guides/account-types-and-billing/accounts-and-pricing-plans#canceling-your-account) by owners. In addition, account owners serve as the primary point of contact for billing purposes. Invoices are sent to them, but if a specific billing contact (/guides/account-types-and-billing/who-receives-your-bill) has been defined for an account, invoices go to that contact instead.

To transfer account ownership to another user, contact support@fastly.com (mailto:support@fastly.com) for assistance.

---

§ Email and password changes (/guides/user-access-and-control/email-and-password-changes)

The Fastly web interface (/guides/basic-concepts/about-the-web-interface-controls) allows you to change the name, email address, and password currently associated with your account.

Changing your name or email address

Follow these instructions to change the name or email address currently associated with your account:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Your profile** link. The Your profile page appears.
3. Fill out the page as follows:
   - In the **Name** field, type your name.
   - In the **Email (login)** field, type your email address.
4. Click **Update Profile** to save the changes.
5. If you've changed your email address, confirm your password in the window that appears.

Changing your password

Follow these instructions to change the password currently associated with your account:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Change password** link. The Change password page appears.
3. Fill out the page as follows:
   - In the **Current password** field, type your existing password.
   - In the **New password** field, type the new password.
   - In the **Confirm password** field, type the new password a second time.
4. Click **Change Password** to save the changes.

Password requirements

When choosing a password keep in mind that it must:

- be at least 7 characters long
- contain at least one letter and one number

In addition, passwords cannot solely contain:

- sequences of letters or numbers (e.g., `12345678`, `abcdefg`)

- repeated characters (e.g., `222222`, `aaaaaa`)
- adjacent key placements on a standard keyboard (e.g., `QWERTY`)

The system will prevent you from choosing a password that:

- matches commonly used passwords (e.g., `password123`, `changeme`)
- uses popular dictionary words in passwords less than 16 characters (e.g., `batterystaple`)
- matches your user name or your email address

---

§ Merging accounts (/guides/user-access-and-control/merging-accounts)

If several employees at your company independently create testing accounts when learning about Fastly services (/guides/detailed-product-descriptions/), you can have those testing accounts merged into a single account by emailing the Customer Support team at support@fastly.com (mailto:support@fastly.com) with the following information:

- the Customer IDs (/guides/account-management-and-security/finding-and-managing-your-account-info) of the accounts to be merged
- which account should be considered the primary account (any other accounts will be merged into the primary)

After you contact us, we'll reach out to verify the ownership of each account. If we can confirm ownership, we'll initiate a merge.

• Guides (/guides/) > Account management > Account management and security (/guides/account-management-and-security/)

§ Account lockouts (/guides/account-management-and-security/account-lockouts)

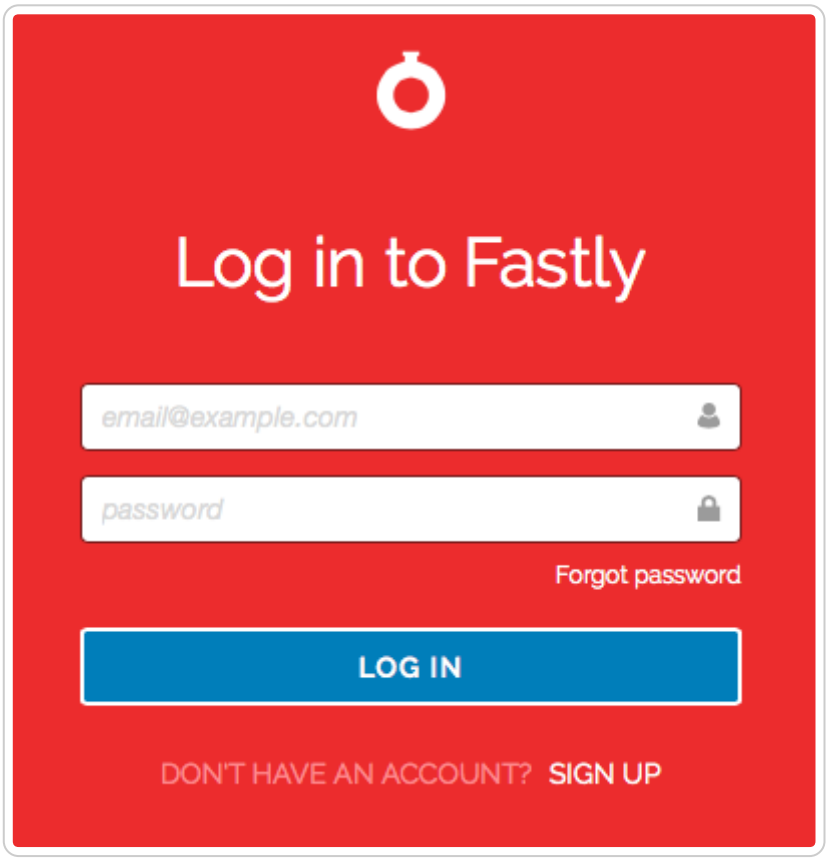
Why is my account locked?

For security reasons, Fastly limits the number of times someone can try logging in to an account. We don't want to give people unlimited attempts at guessing your password, so we stop them from trying after a limited number of failed attempts to sign in. You can change your password (/guides/user-access-and-control/email-and-password-changes#changing-your-password) at any time when you're logged in to your account.

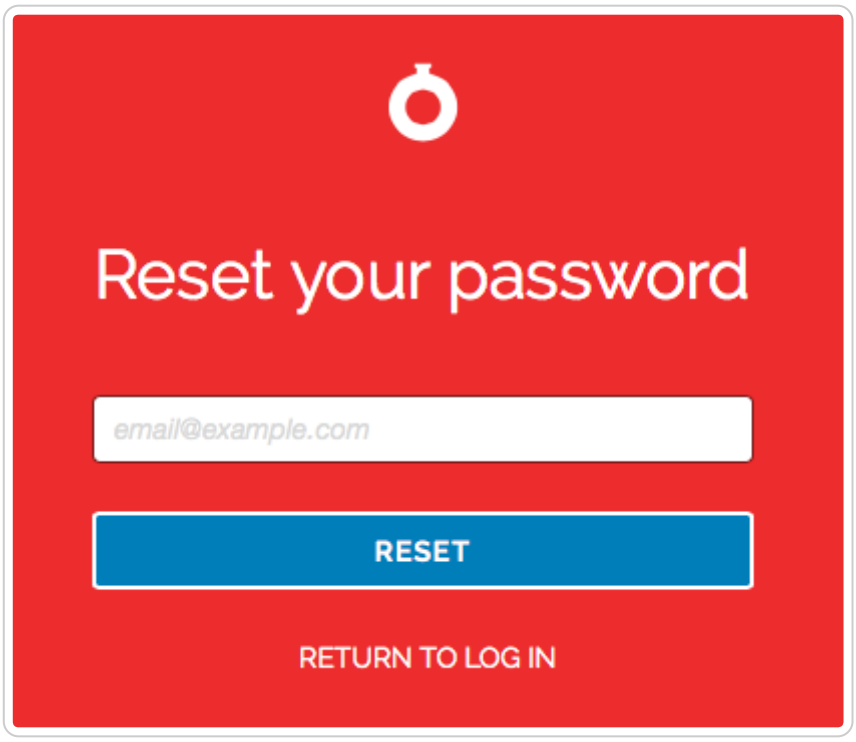
I am not using two-factor authentication. How can I access my account?

Once locked, you will not be able to sign in to your account, even with the correct password. To unlock your account because you exceeded the number of guesses you were allowed:

1. Point any standard web browser (/guides/debugging/browser-recommendations-when-using-the-fastly-application) to the Fastly login page (https://manage.fastly.com). The login controls appear.



2. Click the **Forgot Password** button underneath the password field. The Reset your password page appears.



3. In the **Email (login)** field, type the email address you normally use to log in to your Fastly account, and then click the **Reset** button. Password reset instructions will be emailed to you.
4. Click on the password reset link in the emailed instructions that the system sends you. The Reset Your Password page appears.
5. Click the **Reset Password** button. The system sends you a temporary password to the email address you supplied.
6. Using the temporary password you receive, log in to your account. The controls to create a new password appear.
7. Fill out the **Reset Password** controls as follows:
   - In the **Current Password** field, type the temporary password that the system emailed to you when you requested a password reset.
   - In the **New Password** field, type a new password to replace the temporary password you were sent.
   - In the **Confirm Password** field, type the new password a second time to confirm it.
8. Click the **Change Password** button. The system changes your password and logs you into your account.

I am using two-factor authentication. How can I access my account?

I don't have my mobile device.

If you do not have access to your mobile device, you can complete the login process using one of your recovery codes. These were the recovery codes you saved in a secured location outside of your Fastly account when two-factor authentication was first enabled. You can continue to use your recovery codes until your device is once again accessible. Recovery codes can only be used once, however, so remember to regenerate a new set to avoid running out before you recover your mobile device.

If you don't believe you will be able to recover your lost mobile device and you still have at least two recovery codes left, you can log in with one recovery code and disable two-factor authentication with a second code. Once two-factor authentication is disabled, you can re-enable it with a new mobile device at a later time and regenerate a new set of codes.

I don't have my mobile device and I don't have my recovery codes.

If you don't have your mobile device and didn't save any recovery codes, have another user at your company with the superuser role (/guides/user-access-and-control/configuring-user-roles-and-permissions) contact Customer Support at support@fastly.com (mailto:support@fastly.com). Have them inform Customer Support which user needs assistance with their login. After Customer Support verifies that the request is from a superuser, we will provide them with your recovery code. The superuser will then send you this information and reset your password so that you can access your account.

I don't have my phone, I didn't save my recovery codes, and I am the only superuser for the account.

Contact Customer Support at support@fastly.com (mailto:support@fastly.com). We will verify that you are associated with the company by phone. We will use the contact information located on the company website or under the Fastly account tab. Upon verification, we will send you a recovery code and reset your password.

Was my account compromised?

If a user's account appears to be hacked or phished, we may proactively reset the passwords for the affected accounts to revoke access to the hacker. In these cases, we send an email to the account's real owner (you) with additional information on how to reset the password. If you received one of these emails, follow the instructions in the email.

If you think your account has been hacked or phished, contact Customer Support at support@fastly.com (mailto:support@fastly.com) immediately.

How is a locked account different from a blocked account?

Fastly allows you to restrict who can access your Fastly account based on the IP address of the person attempting to log in. This means that even with the correct login name and password, access to your Fastly account may be blocked if the IP doesn't match your company's list of allowed addresses.

If your company enables this optional IP whitelisting (/guides/account-management-and-security/enabling-an-ip-whitelist-for-account-logins), they must keep the list of restricted IP addresses up to date. Only users with the role of superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions) can make changes to the IP whitelist settings (your account owner is always a superuser), and your account owner must have a valid telephone number on file to do so.

If your IP addresses change after whitelisting is enabled and you forget to update your whitelist configuration, you will be locked out of your account. You will need to contact support@fastly.com (mailto:support@fastly.com) to request that a Customer Support representative contact your account's owner via telephone during Fastly's regular business hours. To protect your account's security, we will not unlock your account based on an email request alone.

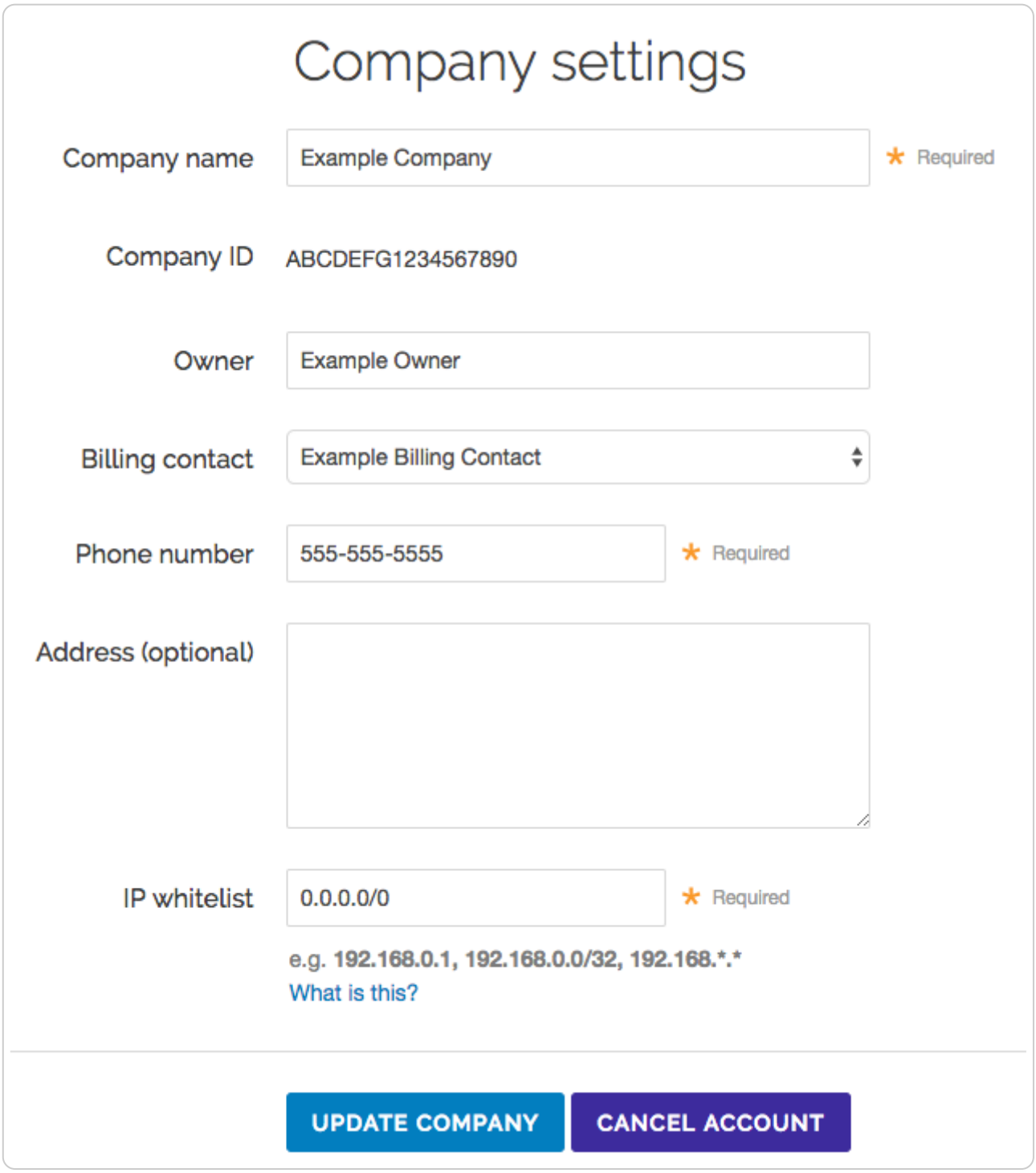
§ Changing your account's company name (/guides/account-management-and-security/changing-your-accounts-company-name)

Fastly allows you to change your account's company name at any time after it's been created.

> ❗ **IMPORTANT:** You must be the account owner (/guides/user-access-and-control/configuring-user-roles-and-permissions#account-ownership-and-how-to-transfer-it) or be assigned the role of superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions) to change your account name.

To change the company name, follow the steps below.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. In the **Company name** field of the **Company settings** area, replace the current company name.



3. Click the **Update Company** button.

§ Enabling an IP whitelist for account logins (/guides/account-management-and-security/enabling-an-ip-whitelist-for-account-logins)

Fastly allows you to define the range of IP addresses authorized to access your Fastly account. This optional IP whitelisting functionality is not enabled by default.

> ⚠ **WARNING:** ***If you decide to use optional IP whitelisting, your account owner must have a valid telephone number on file.*** During setup, Fastly checks your current IP address against the list you provide to ensure you don't lock yourself out of your account. If your IP addresses change at a later date (for example, because you move offices) and you forget to update your whitelist configuration, you will be locked out of your account. You will need to contact support@fastly.com (mailto:support@fastly.com) to request that a Customer Support representative contact your account's owner via telephone during Fastly's regular business hours. To protect your account's security, ***we will not unlock your account (/guides/account-management-and-security/account-lockouts) based on an email request alone***.

Enabling an IP whitelist

To restrict access to your Fastly account based on a specific list or range of IP addresses, follow these steps.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. In the **IP whitelist** field of the **Company settings** area, replace `0.0.0.0/0` (the default IP range indicating no whitelisting) with the IP addresses to which your account access should be restricted.



   In the IP whitelist field you can include single or multiple IP addresses or IP ranges (separated by commas) as follows:

   - a single IPv4 address (e.g., replace the default with `192.168.0.1`)
   - an IPv4 CIDR range (e.g., replace the default with `192.168.0.0/32`)
   - an IPv4 Wildcard range (e.g., replace the default with `192.168.0.*`, `192.168.*.1`, `192.168.*.*`)

3. Click the **Update Company** button.

Disabling an IP whitelist

To disable IP whitelisting on your Fastly account, follow these steps.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. In the **IP whitelist** field of the **Company settings** area, type `0.0.0.0/0` (the default IP range indicating no whitelisting).
3. Click the **Update Company** button.

---

§ Enabling and disabling two-factor authentication (/guides/account-management-and-security/enabling-and-disabling-two-factor-authentication)

Fastly supports two-factor authentication, a two-step verification system, for logging in to the web interface. In a two-factor authentication security process, users provide two means of identifying themselves to the system, typically by providing the system with something they know (for example, their login ID and password combination) and something they have (such as an authentication code). Organizations can enable company-wide two-factor authentication to require all users within the organization to use two-factor authentication.

Before you begin

You'll need to enter an authentication code regularly. Once two-factor authentication has been enabled, an authentication code will be requested upon login at least every 14 days for each computer and browser you use to access the Fastly web interface.

A mobile device is required. Using this security feature with a Fastly account requires a mobile device capable of scanning a barcode or QR code using a downloadable authenticator application. We recommend the following:

- For Android, iOS, and Blackberry: Google Authenticator (https://support.google.com/accounts/answer/1066447?hl=en)
- For Android and iOS: Duo Mobile (https://guide.duo.com/third-party-accounts)

- For Windows Phone: Authenticator (https://www.microsoft.com/en-us/store/p/microsoft-authenticator/9nblgggzmcj6)

There are special requirements for using this feature with API tokens. See the API token documentation (/api/auth#two-factor-authentication) for more information.

Managing two-factor authentication as a user

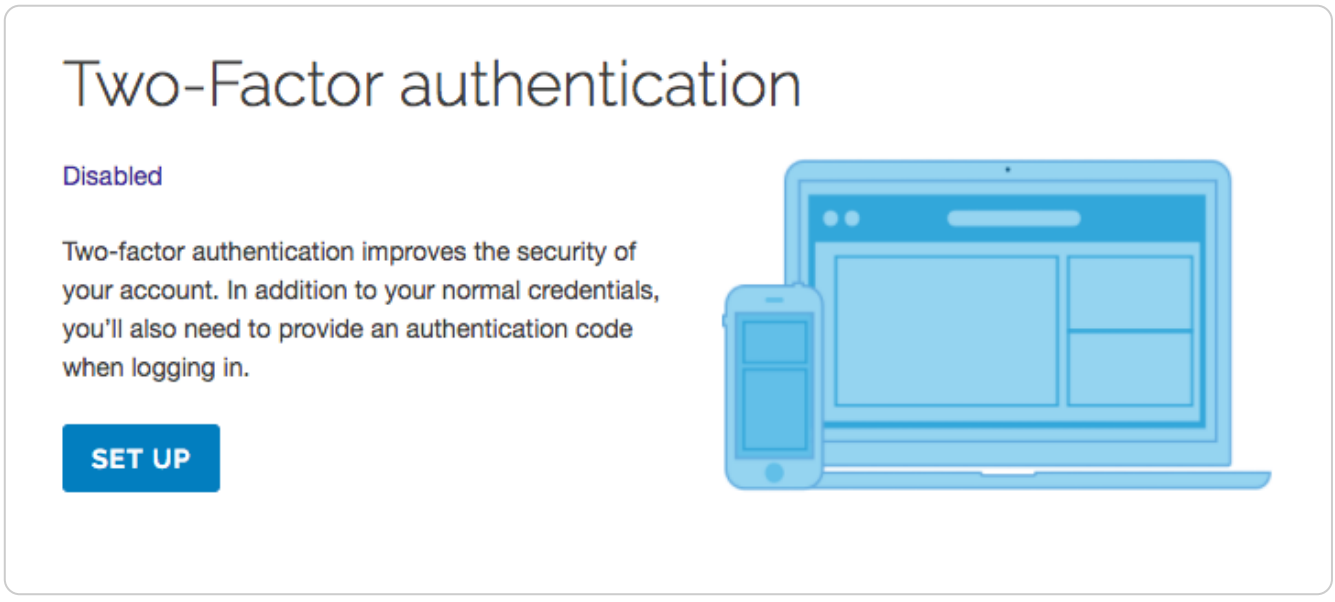
Depending on whether or not your organization has enabled company-wide two-factor authentication, you may be able to enable and disable two-factor authentication for your personal account. We also have instructions for recovering access to your account if you lose your mobile device.

Enabling two-factor authentication

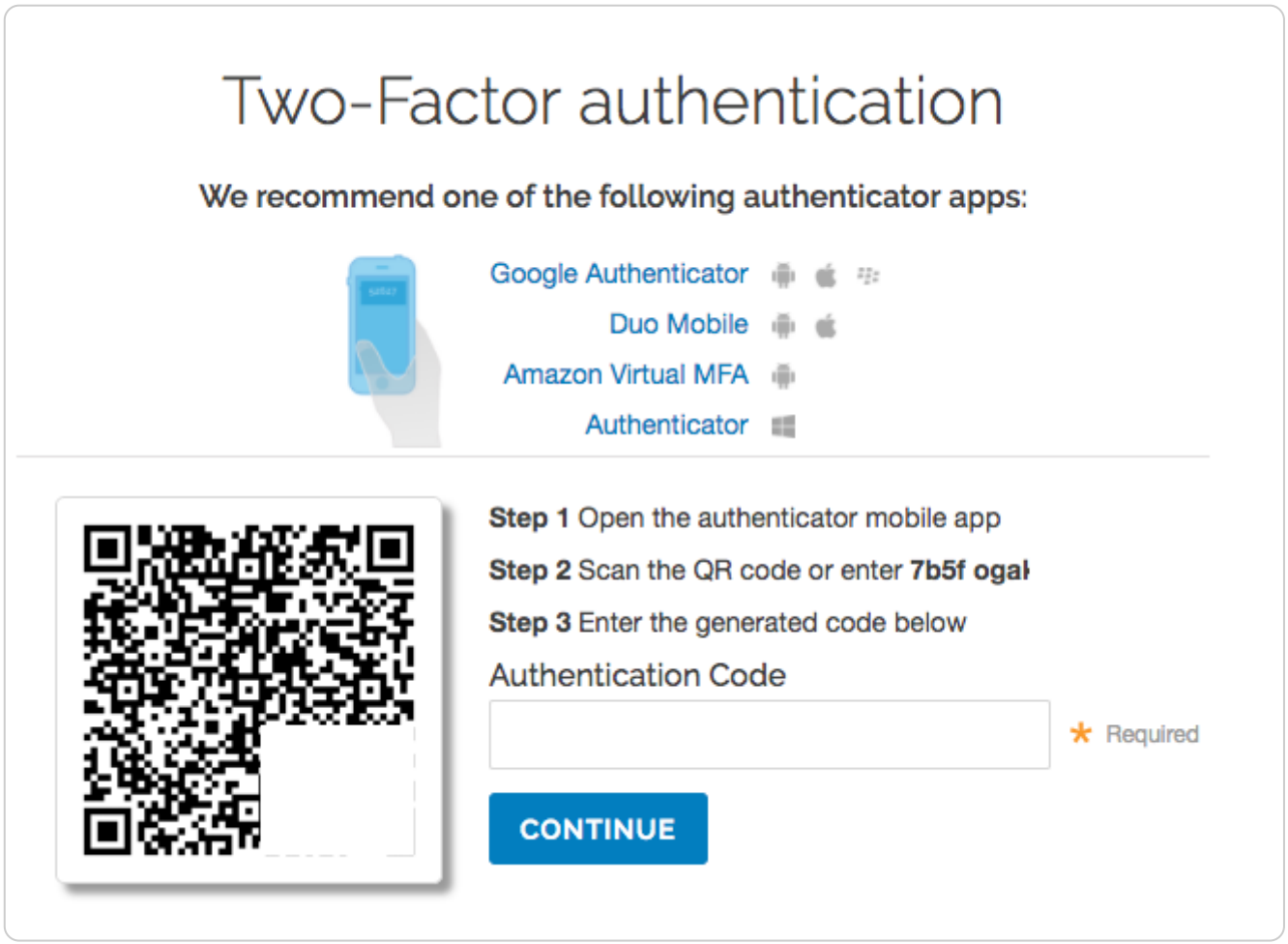
To enable two-factor authentication for your user account, follow the steps below.

> ❗ **IMPORTANT:** If your organization has enabled company-wide two-factor authentication, you will be required to set up two-factor authentication when you log in to the Fastly web interface. Skip to step six for instructions.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Two-factor authentication** link. The Two-Factor authentication page appears.



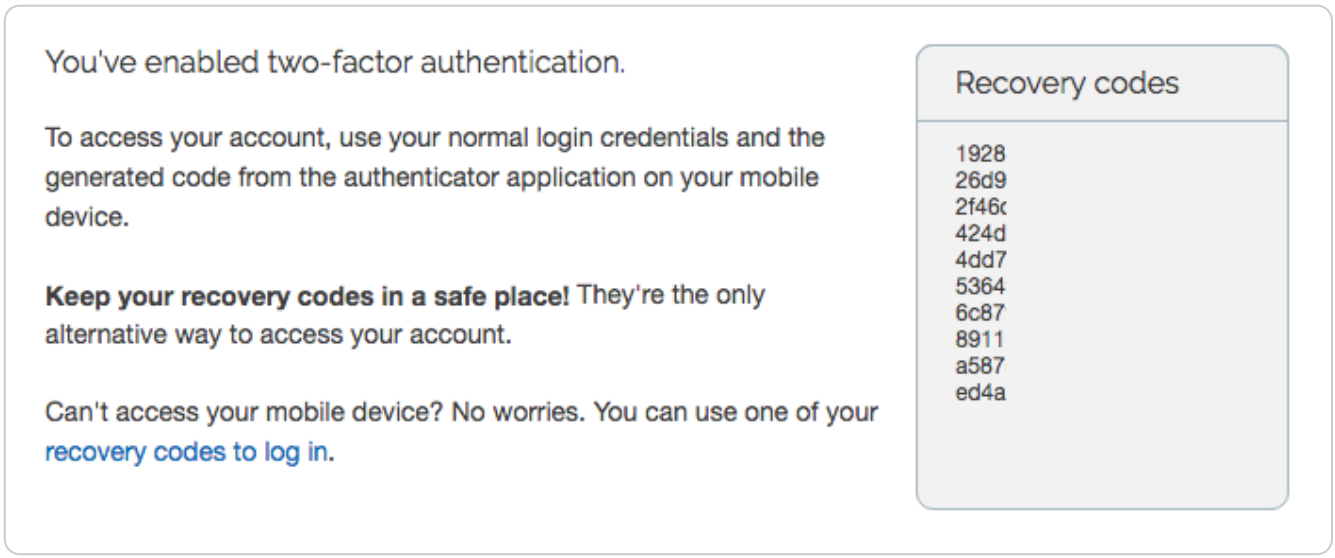
3. Click the **Set Up** button. The password verification screen appears.
4. Verify your Fastly password and then click **Continue**. The authentication QR code appears.



IMPORTANT: The QR code above is an example. Scan the one that appears in the Fastly application, not in this guide.

5. Launch the authenticator application installed on your mobile device and scan the displayed QR code or manually enter the key displayed in the setup window. A time-based authentication code appears on your mobile device. Depending on your device, however, a browser link may first appear. You need to click this link to save it. When you do, the words `Secret saved` appear briefly.
6. In the **Authentication Code** field in the Fastly application, type the time-based authentication code displayed on your mobile device.

   ANDROID USERS: A common time syncing issue may cause your authenticator codes to fail. You can correct this using Google's instructions (https://support.google.com/accounts/answer/185834?hl=en#sync) for your authenticator application.
7. Click **Continue**. The confirmation screen appears along with your recovery codes.



IMPORTANT: If you're ever unable to access your mobile device, the displayed recovery codes can be used to log in when your account has two-factor authentication enabled. Each of these recovery codes can only be used once, but you can regenerate a new set of 10 at any time (any unused codes at that time will be invalidated). Store your recovery codes in a safe place.

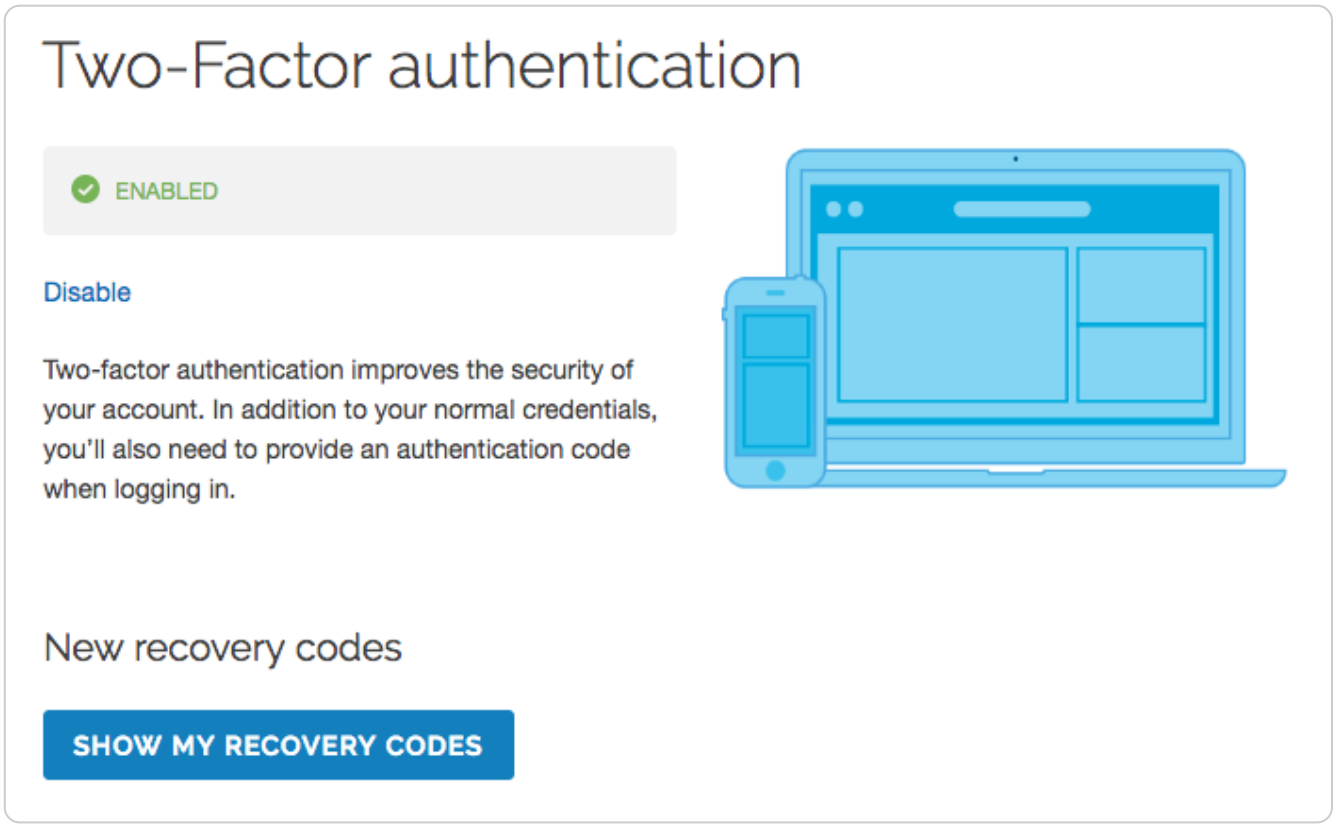
After you enable two-factor authentication, logging in to your Fastly account will require your email address and password, and then an authentication code generated by the authenticator application you've installed on your mobile device. By default, the system requires you to authenticate your login using an authentication code at least every two weeks for each computer and browser you use to access the Fastly web interface.

Disabling two-factor authentication

Once two-factor authentication is enabled for your account, you can disable it at any time by following the steps below.

IMPORTANT: If your organization has enabled company-wide two-factor authentication, you cannot disable two-factor authentication for your account.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Two-factor authentication** link. The Two-Factor authentication page appears.



3. Click **Disable**. The verification screen appears.
4. In the **Authentication Code** field, type the time-based authentication code displayed in the authenticator application on your mobile device, then click **Confirm and Disable**.

> ℹ **NOTE:** If you have lost your mobile device, you can enter a recovery code in the **Authentication Code** field. For more information, see the section on what to do if you lose your mobile device.

What to do if you lose your mobile device

If you lose your mobile device after enabling two-factor authentication, use a recovery code to log in to your Fastly account. You can continue to use recovery codes to log in until you get your mobile device back. Recovery codes can only be used once, however, so remember to regenerate a new list of codes to avoid running out before you recover your mobile device.

If you do not believe you will be able to recover your lost mobile device and you still have at least two recovery codes left, you can log in with one recovery code and disable two-factor authentication with a second code. Once two-factor authentication is disabled, you can re-enable it with a new mobile device at a later time and regenerate a new set of codes.

If your organization has enabled company-wide two-factor authentication, you can contact a superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions) for your organization and ask them to reset your two-factor authentication.

Locked out of your account? See our article on what you can do about it (/guides/account-management-and-security/account-lockouts).

Managing two-factor authentication as a superuser

If you are assigned the superuser role (/guides/user-access-and-control/configuring-user-roles-and-permissions) for your organization, you can view who has two-factor authentication enabled the User management settings for your Account. Users with this feature enabled have padlocks displayed next to their names.



To disable two-factor authentication for any user within your organization, select **Disable 2FA** from the menu that appears when you click the gear icon next to that user's name.

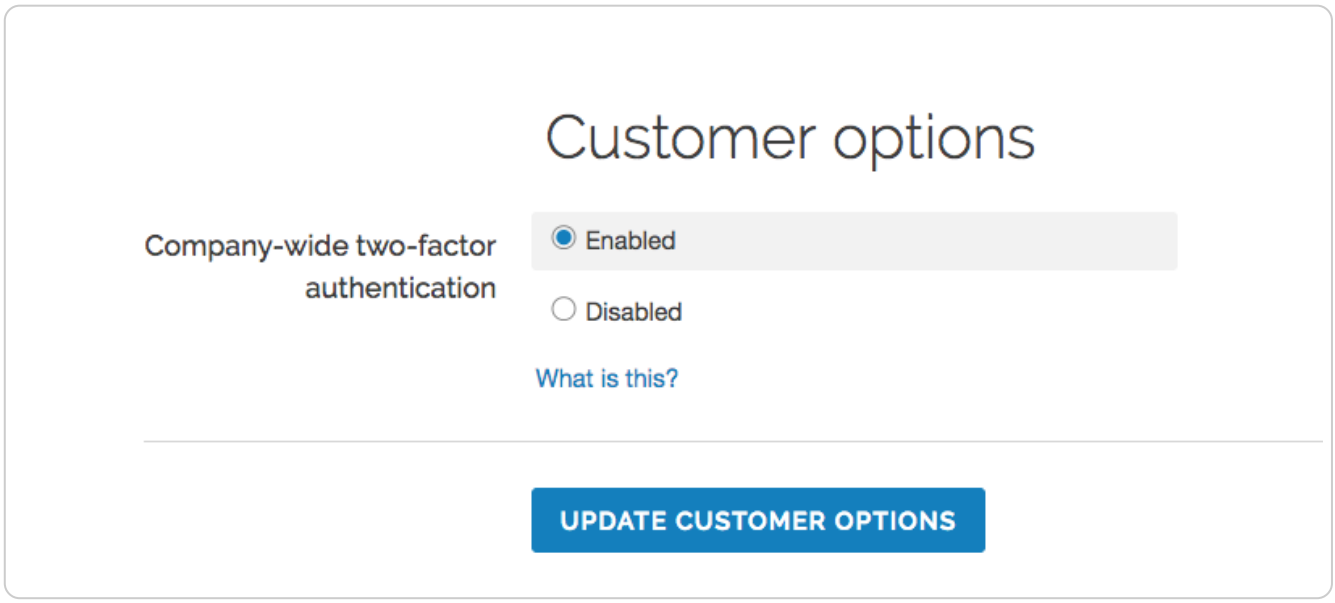
Managing two-factor authentication as a company

Organizations can enable two-factor authentication for all of their users. When the company-wide two-factor authentication feature is enabled, all users within the organization are required to use two-factor authentication to log in to the Fastly web interface, and they cannot disable two-factor authentication for their accounts.

Enabling company-wide two-factor authentication

Users assigned the superuser role (/guides/user-access-and-control/configuring-user-roles-and-permissions) can enable this feature on the Account page. To enable company-wide two-factor authentication for all users within your organization, follow the steps below.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. In the **Customer options** area, select **Enabled** from the **Company-wide two-factor authentication** controls.



3. Click **Update Customer Options**. A warning message appears.
4. Click **Continue**. You will be logged out of the Fastly web interface. This completes the setup process for company-wide two-factor authentication.

Users who have not already enabled two-factor authentication for their accounts will be prompted to do so the next time they log in to the Fastly web interface.

Resetting a user's two-factor authentication

If company-wide two-factor authentication is enabled, and a user within the organization gets locked out of their account or needs to enable a new device, an account superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions) can reset their two-factor authentication. To reset a user's two-factor authentication, follow the steps below.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **User management** link.
3. In the **Users** area, click the gear icon next to a user and then select **Reset 2FA**. A warning message appears.
4. Click **Reset**. The user will need to set up two-factor authentication for their account the next time they log in.

Disabling two-factor authentication for a single user's account

If company-wide two-factor authentication is enabled, a superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions) can disable two-factor authentication for a single user's account. This is typically done for user accounts being used for scripts and session

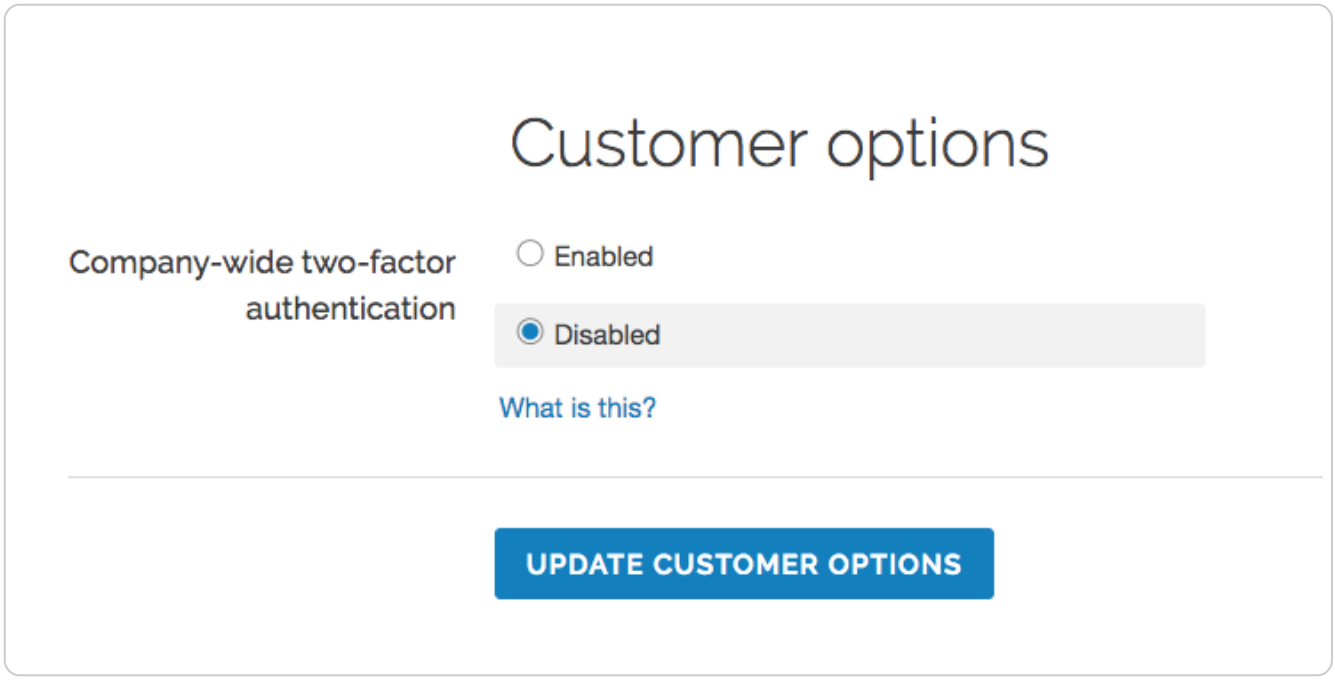
authentication. To disable two-factor authentication for a single user's account, follow the steps below.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **User management** link.
3. In the **Users** area, click the gear icon next to a user and then select **Ignore 2FA**. A warning message appears.
4. Click **Ignore**. Two-factor authentication will no longer be required for the selected user.

Disabling company-wide two-factor authentication

A superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions) can disable company-wide two-factor authentication. Once this feature is disabled, existing users within the organization will be able to manage their own two-factor authentication settings, and new users will not be required to set up two-factor authentication to log in to the Fastly web interface. To disable company-wide two-factor authentication, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. In the **Customer options** area, select **Disabled** from the **Company-wide two-factor authentication** controls.



3. Click **Update Customer Options**. A warning message appears.

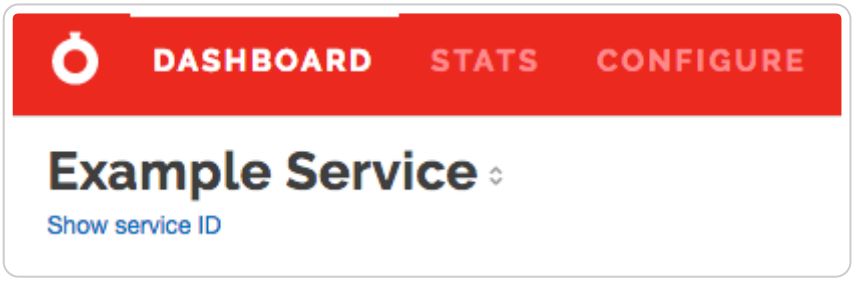
4. Click **Continue**. Company-wide two-factor authentication becomes disabled.

§ Finding and managing your account info (/guides/account-management-and-security/finding-and-managing-your-account-info)

Account information, including your service ID and your customer ID (also called your company ID) can be accessed directly from the Fastly web interface (/guides/basic-concepts/about-the-web-interface-controls).

Finding your service ID

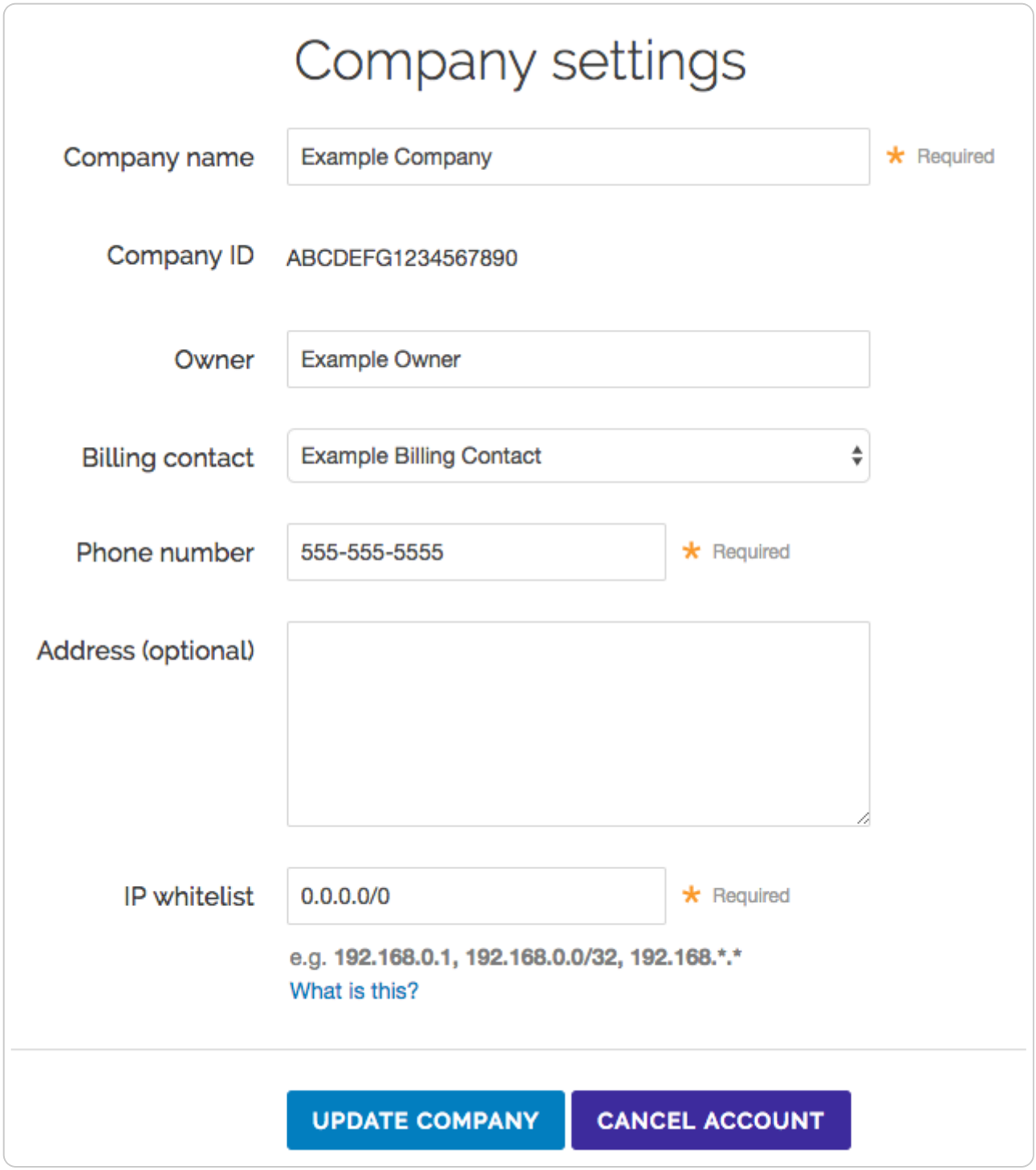
Your **Service ID** appears hidden, immediately below the name of your service on any page. To display the ID, click the **Show service ID** link:



To hide it, click the **Hide service ID** link.

Finding your customer ID

Your **Company ID**, also called your **Customer ID**, appears in the **Company settings** of your **Account** page, which you access by clicking the **Account** link in the user menu:



§ Using API tokens (/guides/account-management-and-security/using-api-tokens)

API tokens are unique authentication credentials assigned to individual users. You need to create an API token to use the Fastly API (/api).

You can use API tokens to grant applications restricted access to your Fastly account and services. For example, an engineer user could limit a token to only have access to a single service, and restrict the scope to only allow that token to purge by URL. Every Fastly user can create up to 100 API tokens.

The API Token Management page (https://manage.fastly.com/account/personal/tokens) allows you to create, view, and delete API tokens associated with your personal account. Superusers (/guides/user-access-and-control/configuring-user-roles-and-permissions) can view and delete any of the API tokens associated with the organization's Fastly account.

> ★ **TIP:** You can also use the Fastly API to create and manage API tokens (/api/auth#tokens).

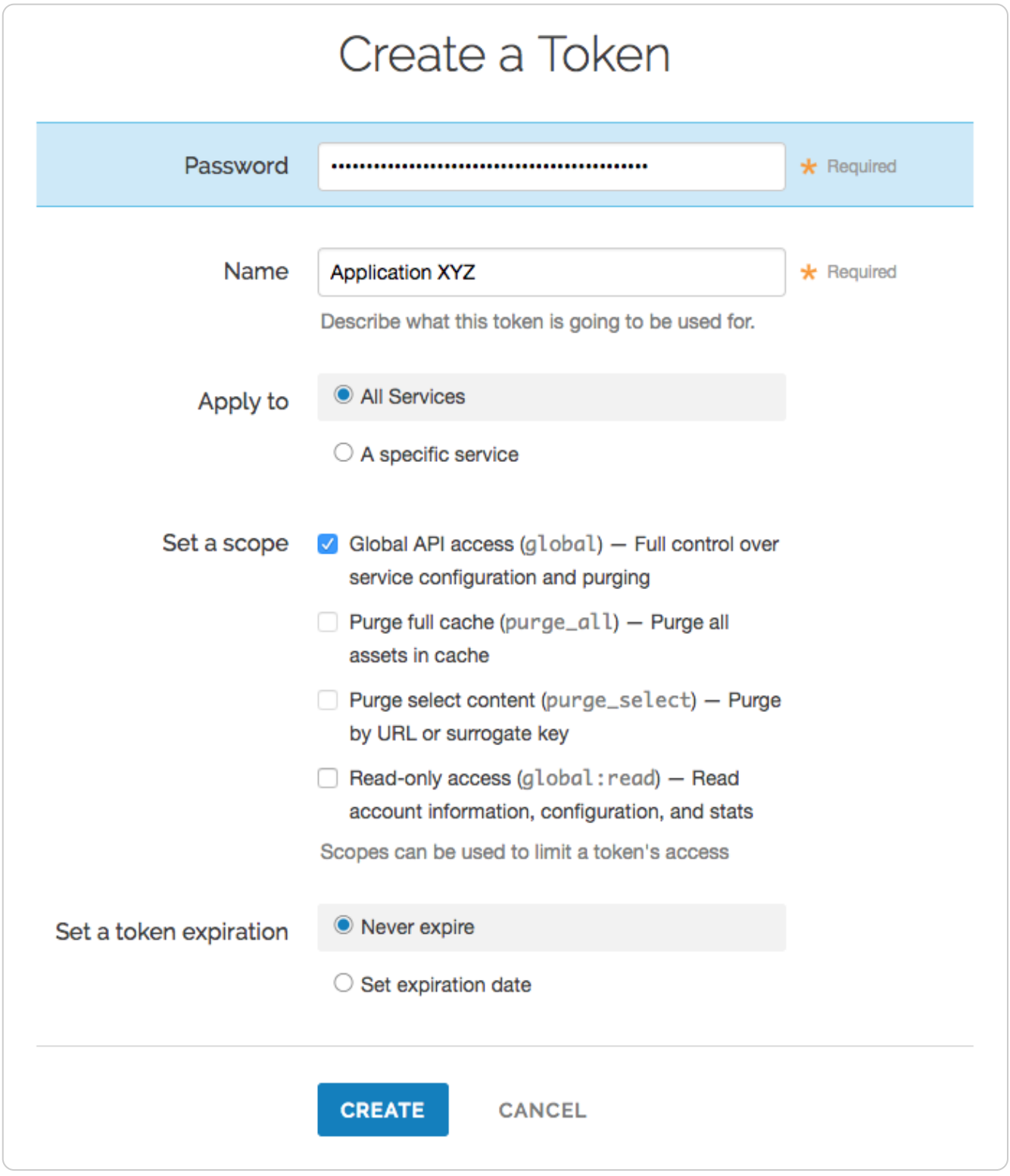
Best practices

Limiting an API token's service access and setting an expiration date restricts a credential's access, which can minimize the risk of damage if a credential is compromised. For more information, review the principle of least privilege (https://en.wikipedia.org/wiki/Principle_of_least_privilege).

Creating API tokens

To create an API token, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Personal API tokens** link. The Personal API Tokens page appears.
3. Click the **Create token** button. The Create a Token page appears.



4. Fill out the **Create a Token** fields as follows:
   - In the **Password** field, type your account password.
   - In the **Name** field, type a descriptive name for the API token that indicates how or where you will to use the token.
   - In the **Apply to** area, optionally select a service to limit the API token to a single service.
   - In the **Set a scope** area, select one or more checkboxes to set a token's scope:

        - **Global API access (global):** Allows access to all endpoints, including purging.
        - **Purge select content (purge_select):** Allows purging with surrogate-key and URL. Does not include the ability to purge all cache.
        - **Purge full cache (purge_all):** Allows purging an entire service via purge_all (/api/purge#purge_bee5ed1a0cfd541e8b9f970a44718546) API request.
        - **Read-only access (global:read):** Allows read-only access to account information, configuration, and stats.
    - In the **Set a token expiration** area, optionally set the API token to expire on a specified date. After a token expires, using it for any request will return an HTTP 401 response.
5. Click the **Create** button to create the new API token. The string that comprises the token appears.

This is the credential you'll use to authenticate via the Fastly API. Copy this string to a secure location — it will never be visible again. You may use the same token for multiple applications.

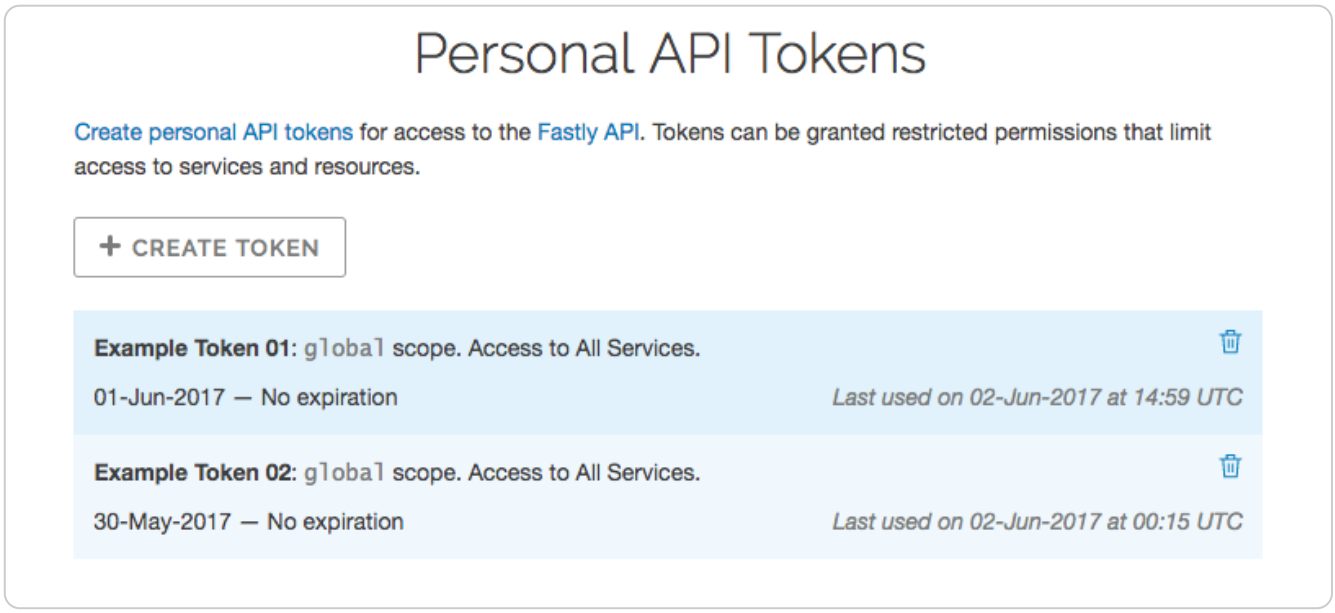
Viewing API tokens

You can view two types of API tokens for your account depending on your assigned role (/guides/user-access-and-control/configuring-user-roles-and-permissions).

Viewing personal API tokens

To view personal API tokens, follow these steps:

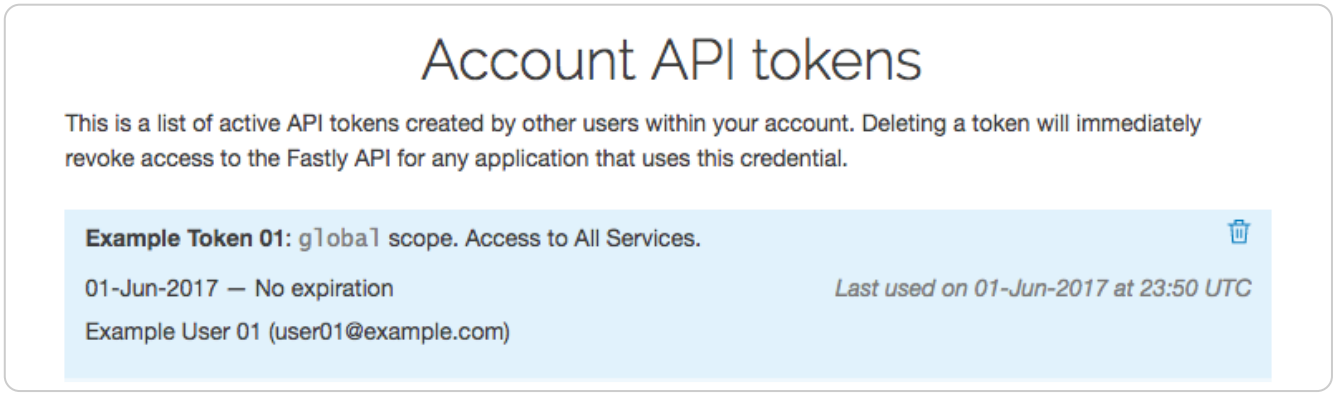
1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Personal API tokens** link. The Personal API tokens page appears with a list of your personal tokens.



Viewing account API tokens

To view account API tokens as a superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions), follow these steps:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click **Account API tokens**. The Account API Tokens page appears with a list of tokens associated with your organization's Fastly account.



Deleting API tokens

⚠ **WARNING:** Deleting an API token will break any integration actively using that credential. Verify you have changed the API token for your integrations before proceeding.

Deleting personal API tokens

To delete a personal API token, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Personal API tokens** link. The Personal API tokens page appears with a list of your personal tokens.
3. Find the API token you want to delete and click the trash icon. A warning message appears.
4. Click the **Delete** button to permanently delete the API token.

Deleting account API tokens

To delete an account API token or to revoke another user's API token as a superuser (/guides/user-access-and-control/configuring-user-roles-and-permissions), follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **Account API tokens**. The Account API Tokens page appears with a list of tokens associated with your organization's Fastly account.
3. Find the API token you want to delete and click the trash icon. A warning message appears.
4. Click the **Delete** button to permanently delete the API token.

Legacy API keys

If you created a Fastly account before May 15th, 2017, you may have used an API key (or multiple API keys) to authenticate API requests. This account-level credential was migrated to a personal API token with a `global` scope and access to all of your services. Because all tokens need to be owned by a user, this credential was assigned to a newly created, synthetic user with the name `Global API Token`.



• Guides (/guides/) > Account management > Account types and billing (/guides/account-types-and-billing/)

§ Accounts and pricing plans (/guides/account-types-and-billing/accounts-and-pricing-plans)

Types of accounts and plans

Fastly offers a variety of accounts and pricing plans, which we detail below. To estimate your monthly charges using our pricing estimator, see our pricing page (https://www.fastly.com/pricing).

Free developer trials

We offer a development trial that allows you to test our services free of charge. We allow you to test up to $50 of traffic for free to ensure everything fits your requirements. Simply sign up for a trial (https://www.fastly.com/signup) and begin testing.

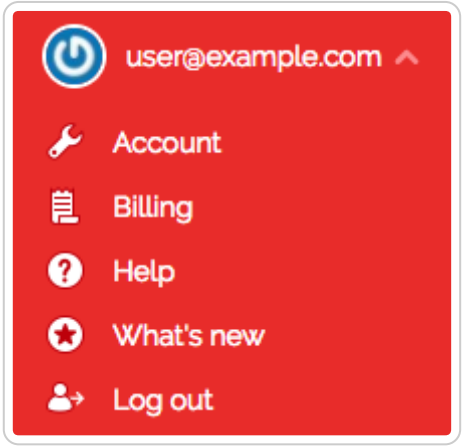
We don't limit functionality during your trial. Once your testing is complete and you're ready to start pushing production traffic our way, you can switch your account to a paid account.

Paid accounts without contracts

After your trial period ends, you can use Fastly's services on a month-to-month basis without having to sign a contract. Be sure you've provided us with your current billing address (/guides/account-types-and-billing/paying-your-bill#changing-your-tax-or-billing-address) as well as your credit card information (/guides/account-types-and-billing/paying-your-bill#changing-your-credit-card-information).

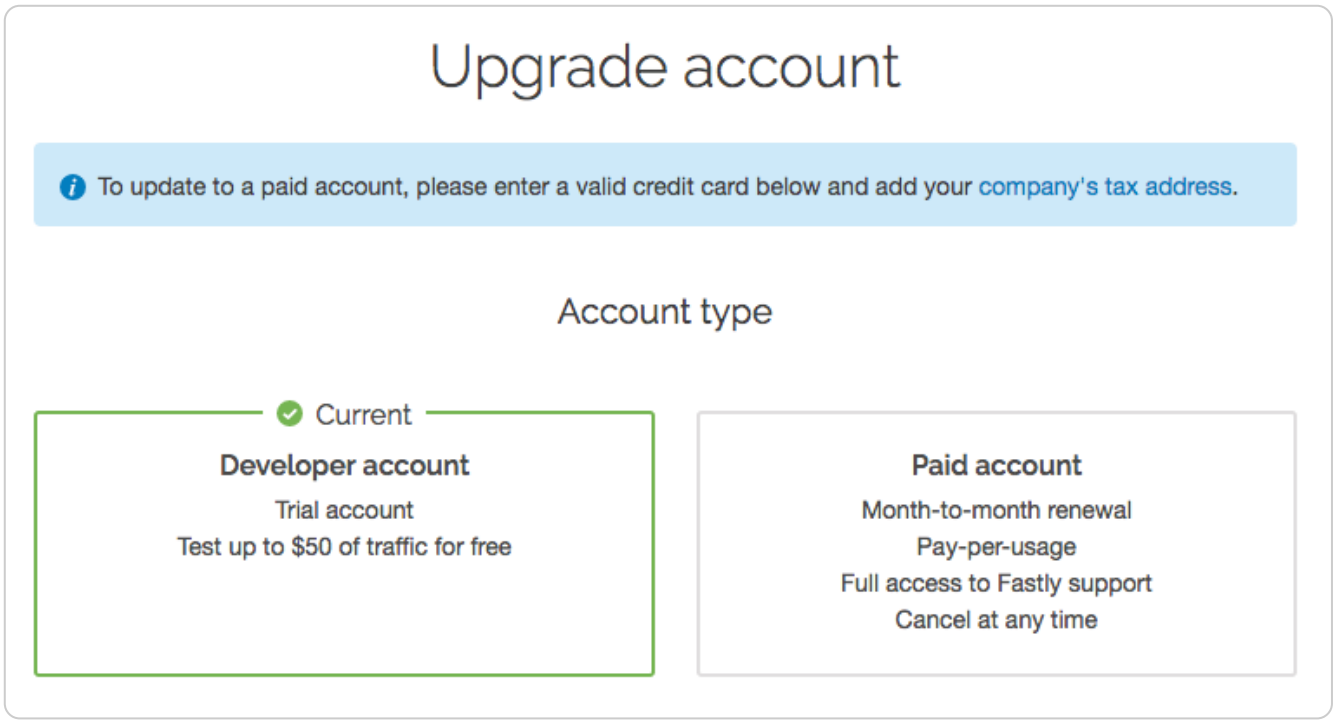
To switch from a developer trial to a paid account without a contract, follow the steps below:

1. Log in to the Fastly web interface.
2. From the user menu, select **Billing**.



Your account's billing information appears.

3. Click the **Upgrade account** link. Information about your plan's current account type appears.



4. Click the **Paid account** plan option.
5. Agree to Fastly's Terms of Service (https://www.fastly.com/terms) by selecting the **I agree to the terms of service** checkbox.
6. Click the **Upgrade Account** button. The development trial option disappears.

Once you switch to a paid account, the developer account plan option disappears and we'll begin billing you automatically (/guides/account-types-and-billing/how-we-calculate-your-bill) at the end of every month using your credit card information (/guides/account-types-and-billing/paying-your-bill#changing-your-credit-card-information). You can cancel your paid account at any time.

Paid accounts with contractual commitments

If you plan to push at least 2TB of data per month and require one of our TLS service options (/guides/detailed-product-descriptions/tls-service-options), or if you plan to push a minimum of 4TB of data per month, it might be worthwhile to consider a contract with Fastly. Contact us at sales@fastly.com (mailto:sales@fastly.com) for more information. We also offer solutions targeted to the needs of specific industries.

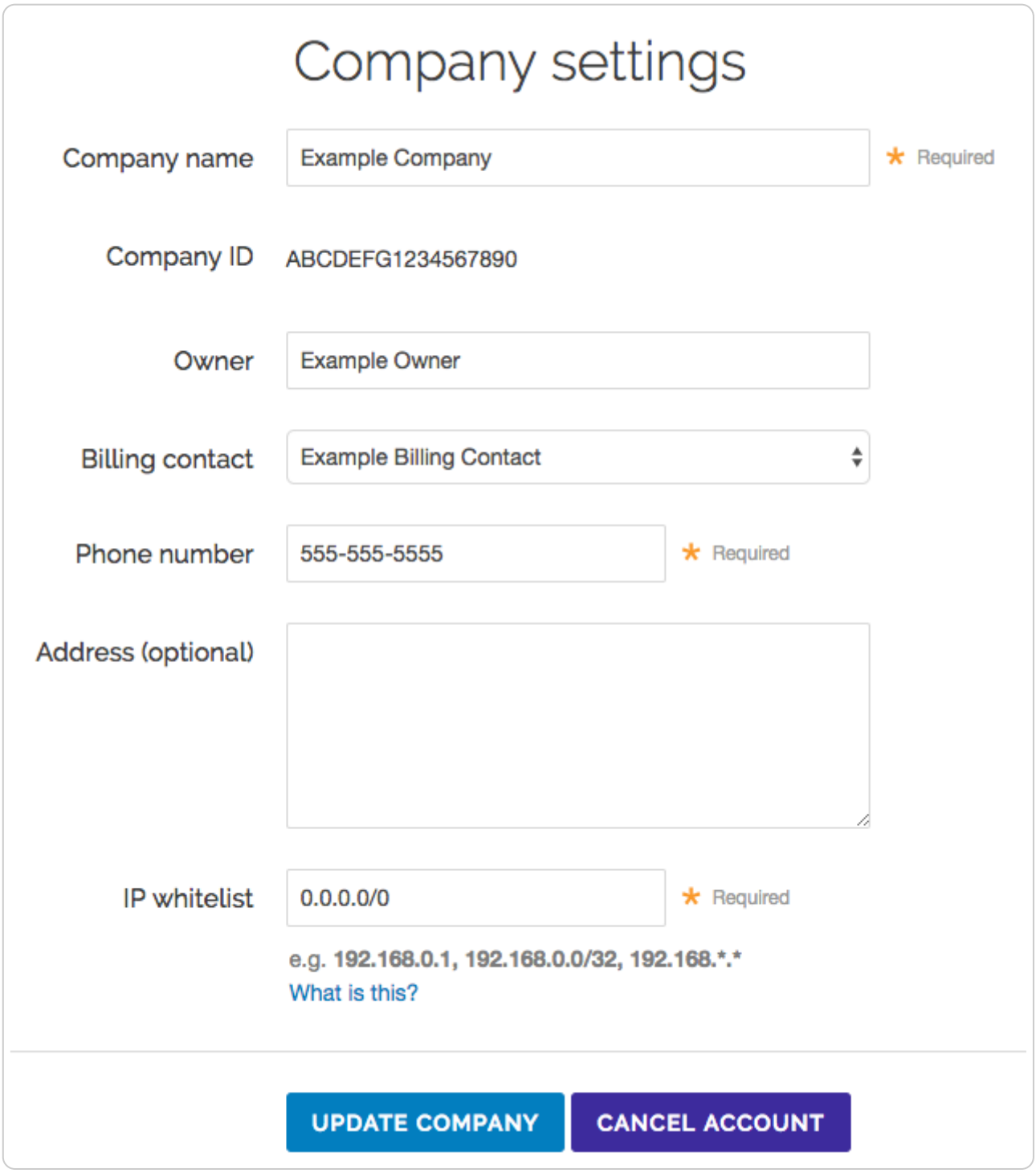
Free open source developer accounts

We're active open source contributors and support the community (https://www.fastly.com/open-source) whenever possible. If you're an open source developer, your bill is on us. Contact us at community@fastly.com (mailto:community@fastly.com) to get started.

Canceling your account

To cancel your account, have your account owner (/guides/user-access-and-control/configuring-user-roles-and-permissions#account-ownership-and-how-to-transfer-it) follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Deactivate (/guides/basic-setup/working-with-services#deactivating-a-service) and then delete all services (/guides/basic-setup/working-with-services#deleting-a-service) on your account.
3. In the **Company settings** area, click the **Cancel Account** button.



A confirmation window appears.

4. In the **Your password** field of the confirmation window, type the password associated with your account and click **Confirm and Cancel**.

After your account is canceled, you'll be billed for any outstanding charges (/guides/account-types-and-billing/how-we-calculate-your-bill) accrued through the day you canceled. For questions about your final billing statement, contact our billing team (mailto:billing@fastly.com) for assistance. If you decide at a later date to reactivate your account, contact Customer Support (mailto:support@fastly.com) and request reactivation.

§ How we calculate your bill (/guides/account-types-and-billing/how-we-calculate-your-bill)

We bill you monthly according to that month's use of Fastly's services. We measure months according to Coordinated Universal Time (UTC). For usage-based charges, bandwidth is recorded in bytes and presented in gigabytes (GB), and requests are recorded individually and presented in units of 10,000.

Fastly uses The International System of Units (http://www.bipm.org/en/publications/si-brochure/chapter3.html) (SI Units) to measure bandwidth. In our calculations, 1 gigabyte (GB) = 10^9 (1,000,000,000) bytes, 1 terabyte (TB) = 10^{12} bytes (or 1,000 GB), and 1 petabyte (PB) = 10^{15} bytes (or 1,000 TB). Your invoice shows your usage and that matches the usage shown on the Stats page (/guides/basic-concepts/about-the-web-interface-controls#about-the-stats-page).

We charge for egress traffic from our POPs (/guides/basic-concepts/fastly-pop-locations), including traffic served to end users and, if shielding is enabled, traffic served from the shield POP to other POPs. Specifically, we charge for each response and for the size of the response (which includes the header and body). Each response is billed as a single request, and the response size in bytes is billed as bandwidth. We charge for bandwidth and requests for content delivered to clients from the CDN and for bandwidth for traffic sent from the CDN to our customers' origins.

> ℹ **NOTE:** If you're using Anycast IP addresses (/guides/basic-configuration/using-fastly-with-apex-domains#anycast-option), these IPs use our global network and will route a request to the nearest POP (https://www.fastly.com/network-map) located in an area that may charge a higher rate. These areas currently include Asia-Pacific, Australia, New Zealand, Latin America, and South Africa. We announce new areas (/guides/basic-concepts/fastly-pop-locations#will-fastly-ever-adjust-pop-locations-or-service-regions-how-will-i-be-notified) regularly via our network status page (https://status.fastly.com/).

Two specific settings related to responses may affect the total charges on your bill. Enabling gzipping (/guides/basic-configuration/enabling-automatic-gzipping) can reduce the size of responses which reduces the bandwidth you use and thus can reduce your total bill. Enabling shielding (/guides/performance-tuning/shielding) may initially result in greater bandwidth use because requests may need to travel between POPs. The reduced load on your origin servers, however, frequently offsets this increased cost and the potential increase in your bill's total.

Charges for any options you've chosen are applied in addition to the bandwidth and request usage we charge for normal content delivery and streaming.

About the monthly minimum charges

We bill a minimum of $50 per month so we can fully support all of our customers. This is the minimum price you'll pay in any month once you've completed your testing trials (/guides/account-types-and-billing/accounts-and-pricing-plans).

For example, say that you're done testing Fastly's services and you've begun to push production-level traffic through Fastly. If most of your site's traffic for the current month is in North America and Europe and your site uses 10GB of traffic over 10 million requests, the combined bandwidth and request charges would be $8.70 for the month. Because this amount falls below the $50 monthly minimum, we would charge you $50 for that month, not $8.70.

Asia-Pacific, Australia, New Zealand, and Latin America prices listed are slightly higher. If most of your site's traffic were in these areas instead of North America and Europe, then at the above traffic levels your bandwidth and request usage charges would still fall below the monthly minimum and we would charge you $50 for that month.

> ℹ **NOTE:** If you're using Fastly for content delivery via Heroku's cloud development services, see Fastly's Heroku add-ons pricing plan (https://elements.heroku.com/addons/fastly) for additional details.

When we charge you for Fastly services

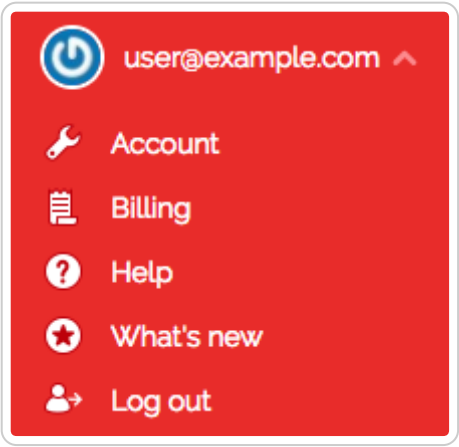
Fastly bills in arrears, not in advance, meaning that we bill you for services after you've used them, not before. For example, if you sign up for and start using Fastly services in January, the bill you receive in February reflects January's charges and services, your March bill reflects February's charges services, and so forth.

How account cancelation affects your bill

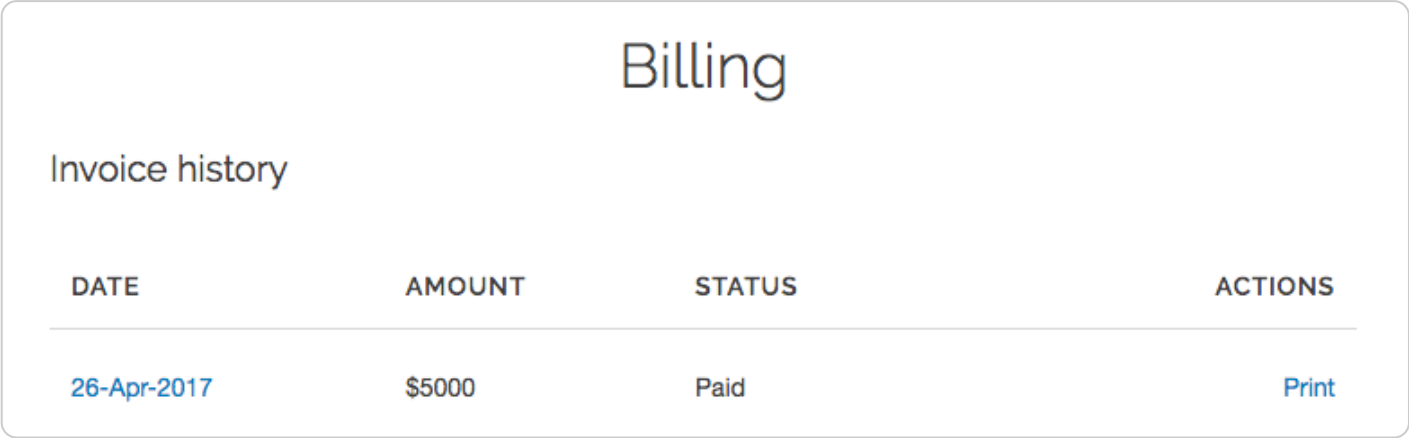
If you ever cancel your account (/guides/account-types-and-billing/accounts-and-pricing-plans#canceling-your-account), you'll be billed for any outstanding charges accrued through the day you canceled, or at least the monthly minimum, whichever amount is greater.

Reviewing the charges to your account

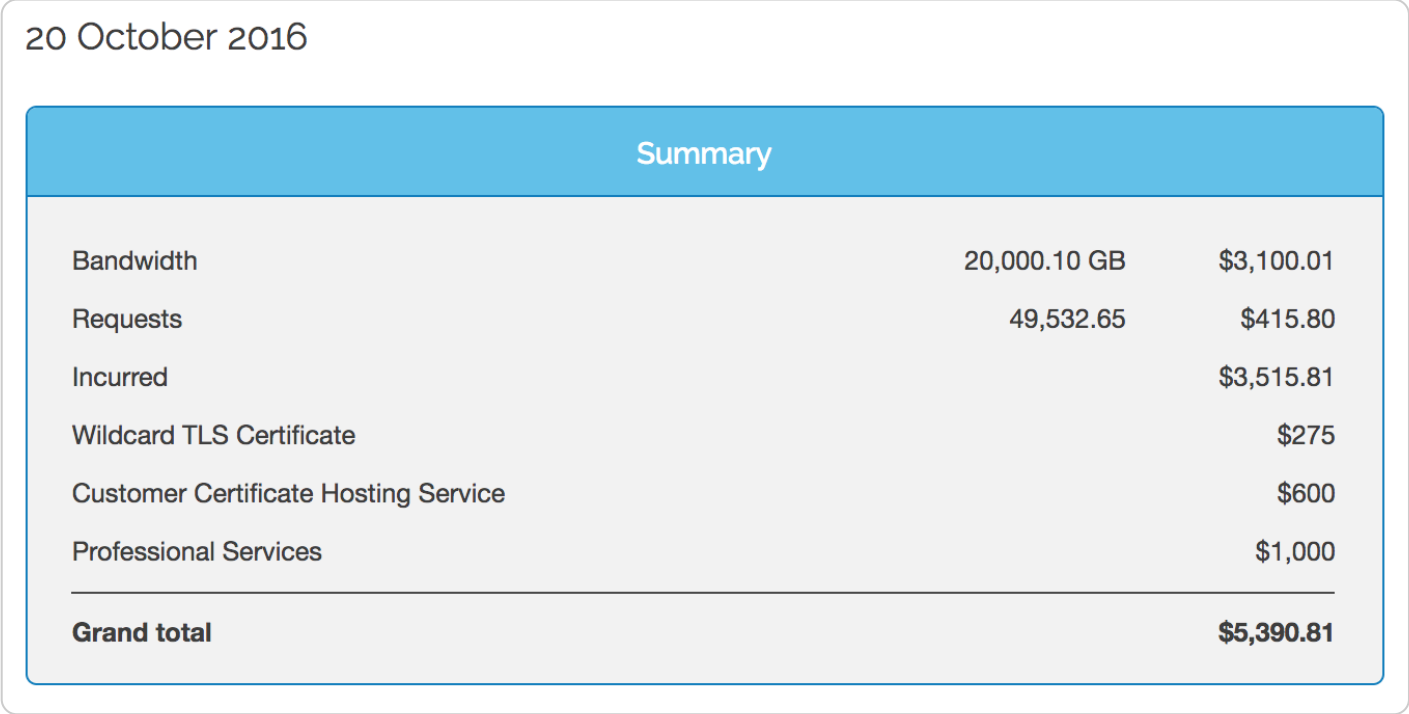
If you've been assigned a superuser or billing role (/guides/user-access-and-control/configuring-user-roles-and-permissions), you can review your account use and the associated charges via the Billing page in the Fastly web interface. Access billing information by selecting Billing from the user menu at the top right of any page.



By default, the current balance for your account appears, followed by the invoice history.

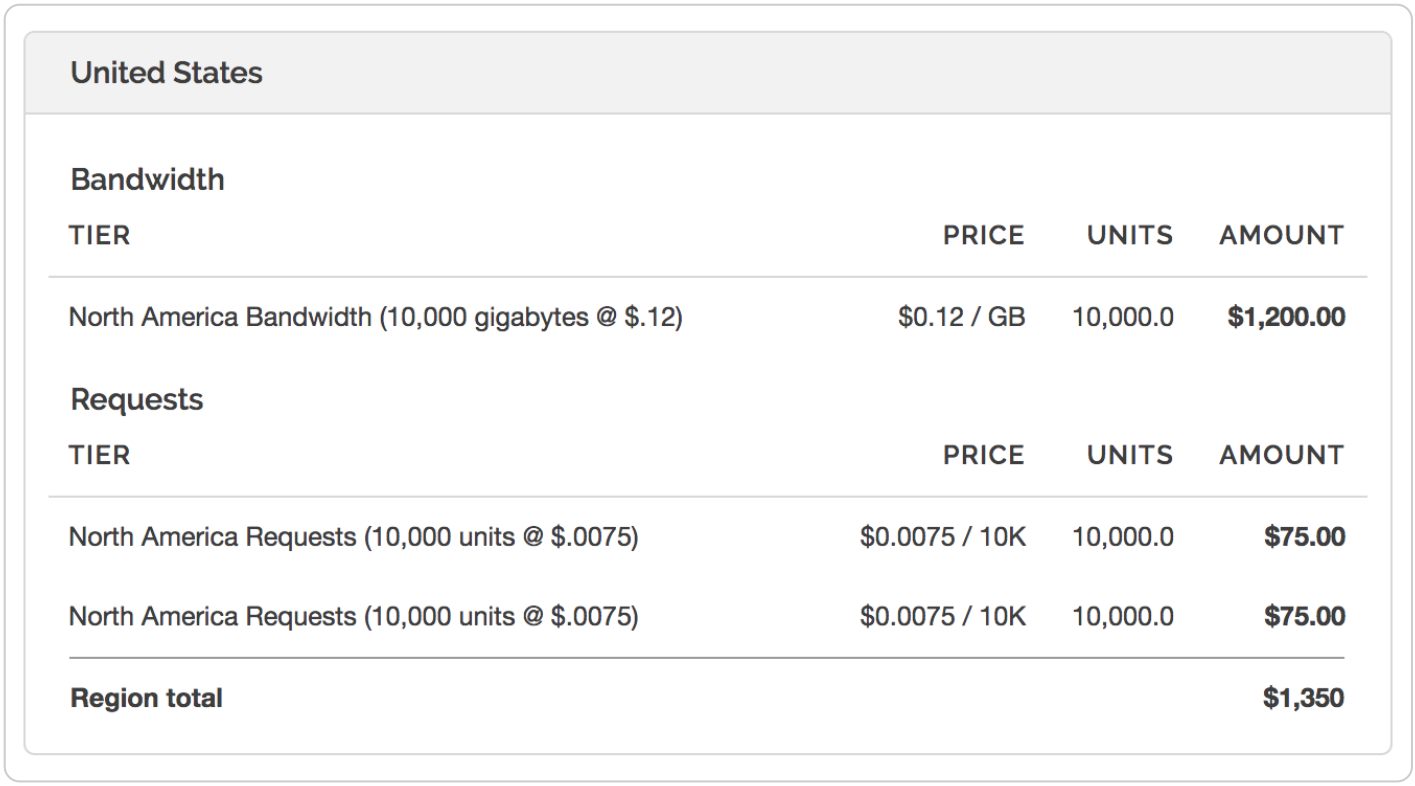


Clicking on the linked date of any invoice displays a summary of charges for that month.



The billing invoice summary includes the overall bandwidth you used and the associated charges, followed by the charges you incurred for requests. The bottom of the summary displays the grand total dollar amount owed for the dated month.

Below the month's summary on the invoice, we include regional bandwidth and request details.



United States

Bandwidth

TIER	PRICE	UNITS	AMOUNT
North America Bandwidth (10,000 gigabytes @ $.12)	$0.12 / GB	10,000.0	**$1,200.00**

Requests

TIER	PRICE	UNITS	AMOUNT
North America Requests (10,000 units @ $.0075)	$0.0075 / 10K	10,000.0	**$75.00**
North America Requests (10,000 units @ $.0075)	$0.0075 / 10K	10,000.0	**$75.00**
Region total			**$1,350**

The bottom of each regional details section includes the total charge for bandwidth and requests for that region alone for the dated month.

Printing account use details

You can print account use details for any month by finding that month in the invoice history and clicking **Print** in the **Actions** column for that month.

§ Paying your bill (/guides/account-types-and-billing/paying-your-bill)

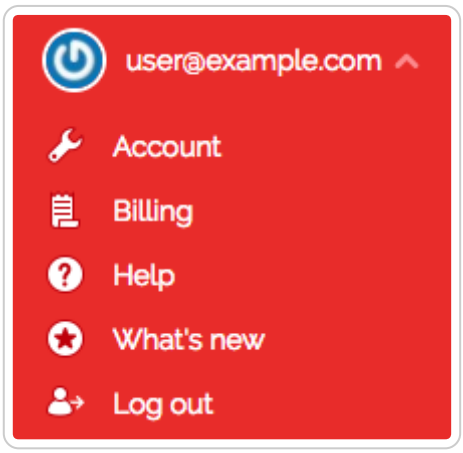
At the end of each month, your account's billing contact (/guides/account-types-and-billing/who-receives-your-bill) will be sent an email summarizing your current usage levels (/guides/account-types-and-billing/how-we-calculate-your-bill) and the charges your account incurred for the month. The email contains a link to an online copy of the related invoice.

You'll need both a valid credit card and current billing address when you switch to a paid, month-to-month account (/guides/account-types-and-billing/accounts-and-pricing-plans#paid-accounts-without-contracts). Once your invoice gets generated, your credit card is automatically charged for the full, outstanding balance.

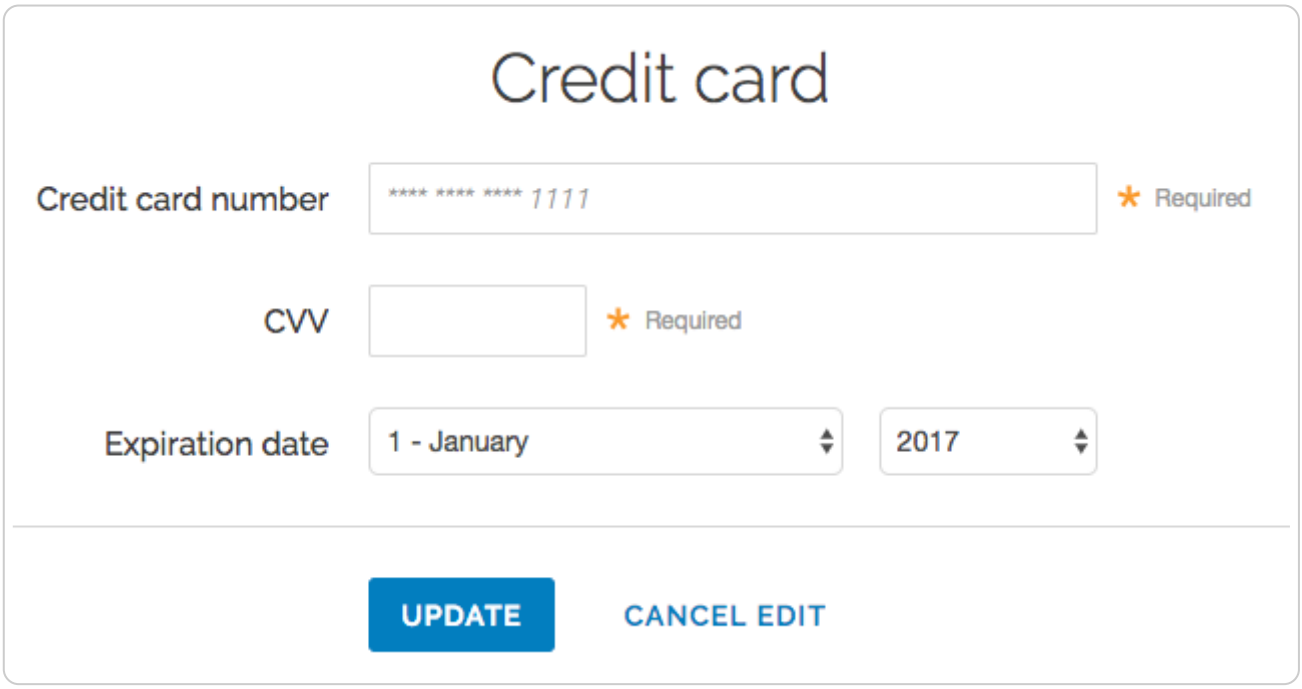
Changing your credit card information

To change the information for the credit card we use for automatic billing, follow the steps below:

1. Log in to the Fastly web interface.
2. From the user menu, select **Billing**.



3. Click the **Credit card** link. The Credit card page appears.
4. Click **Edit**. Details appear for the credit card you have on file with Fastly.
5. Make any necessary changes to the credit card information in the fields provided.



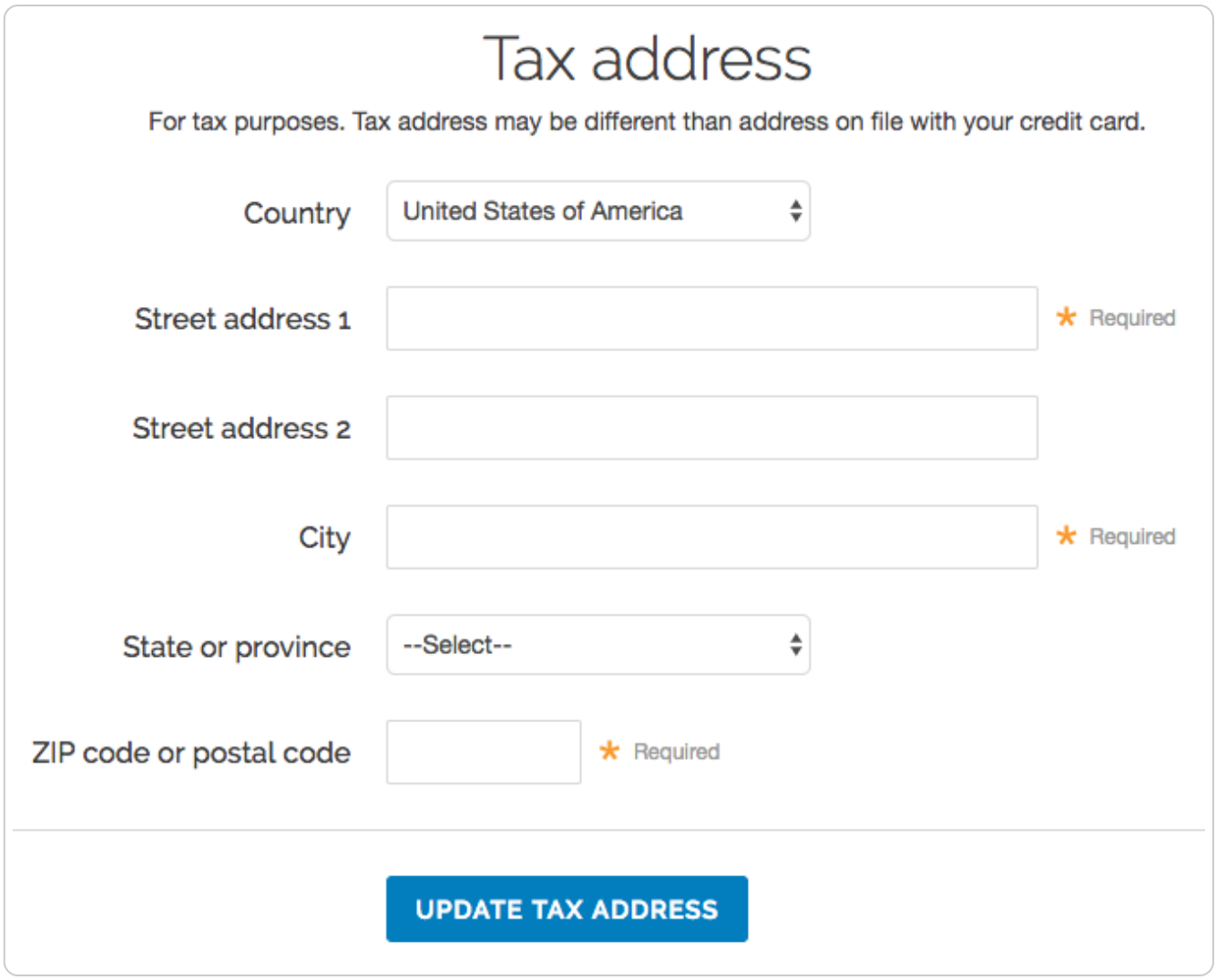
6. Click **Update** to save your credit card information.

★ **TIP:** Fastly never sees your credit card number. All transactions are handled by our fully PCI compliant payment gateway (https://www.vantiv.com/vantage-point/safer-payments/how-tokenization-protects-customer-data), Vantiv.

Changing your tax or billing address

To change your tax or billing address, follow the steps below:

1. Log in to the Fastly web interface.
2. From the user menu, select **Billing**.
3. Click the **Tax address** link and type the tax address information you use in the fields provided.



4. Click the **Update Tax Address** button to save the tax address information.

§ Who receives your bill (/guides/account-types-and-billing/who-receives-your-bill)

By default, your account owner is considered your billing contact and will receive your bill for Fastly services. You can change your billing contact at any time if you've been assigned the superuser role (/guides/user-access-and-control/configuring-user-roles-and-permissions) for an account. If you ever delete your billing contact, billing will automatically revert to the account owner.

❗ **IMPORTANT:** Invoices are only sent to the email addresses of the Account Owner or the Billing Contact. Invoices are not sent to every user assigned a billing role (/guides/user-access-and-control/configuring-user-roles-and-permissions).

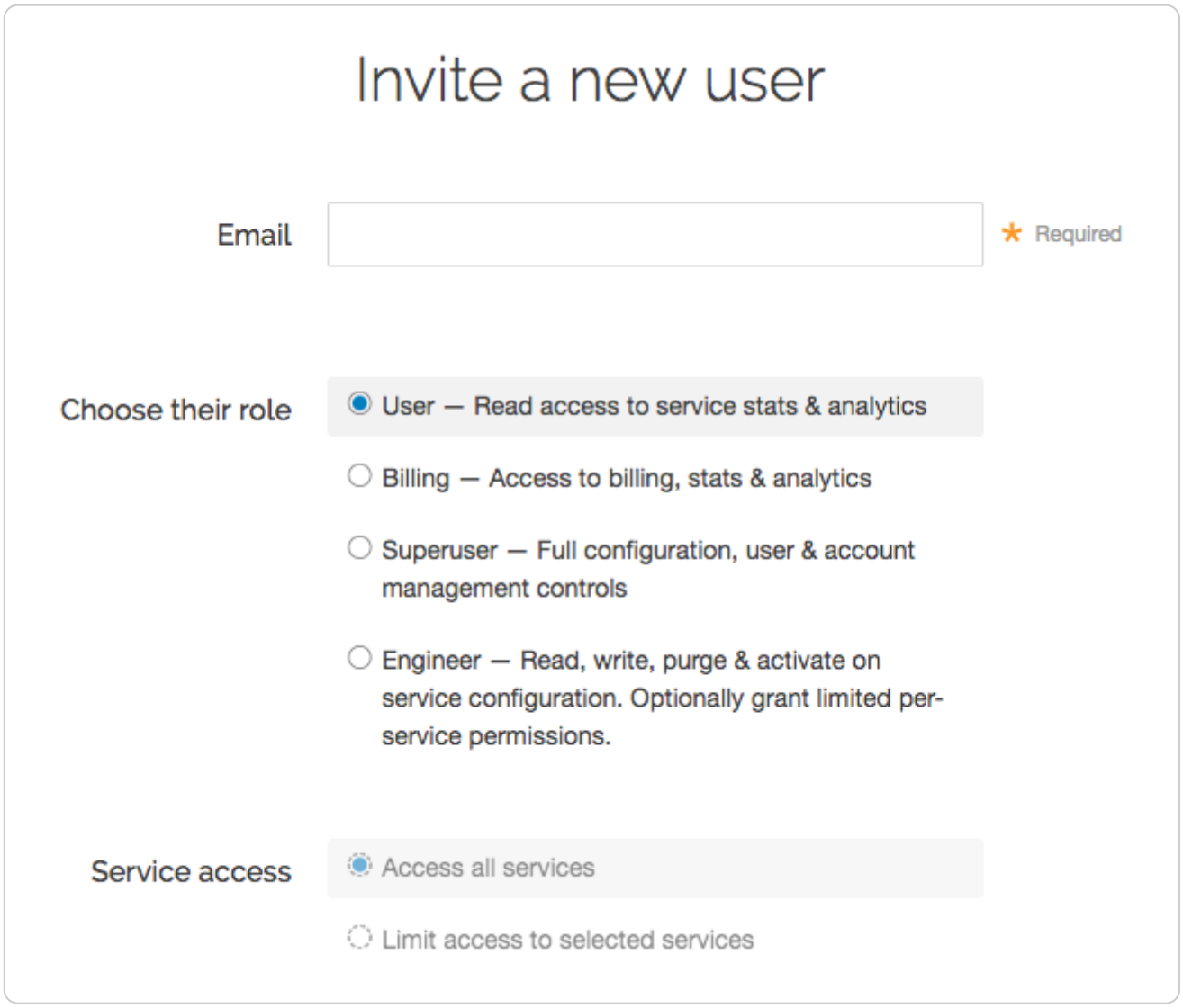
Changing who receives your bill

Follow the steps below to have your billing invoice sent to a person other than the owner of your account.

For new users

To send the billing invoice to a user who has not yet created an account, follow these steps.

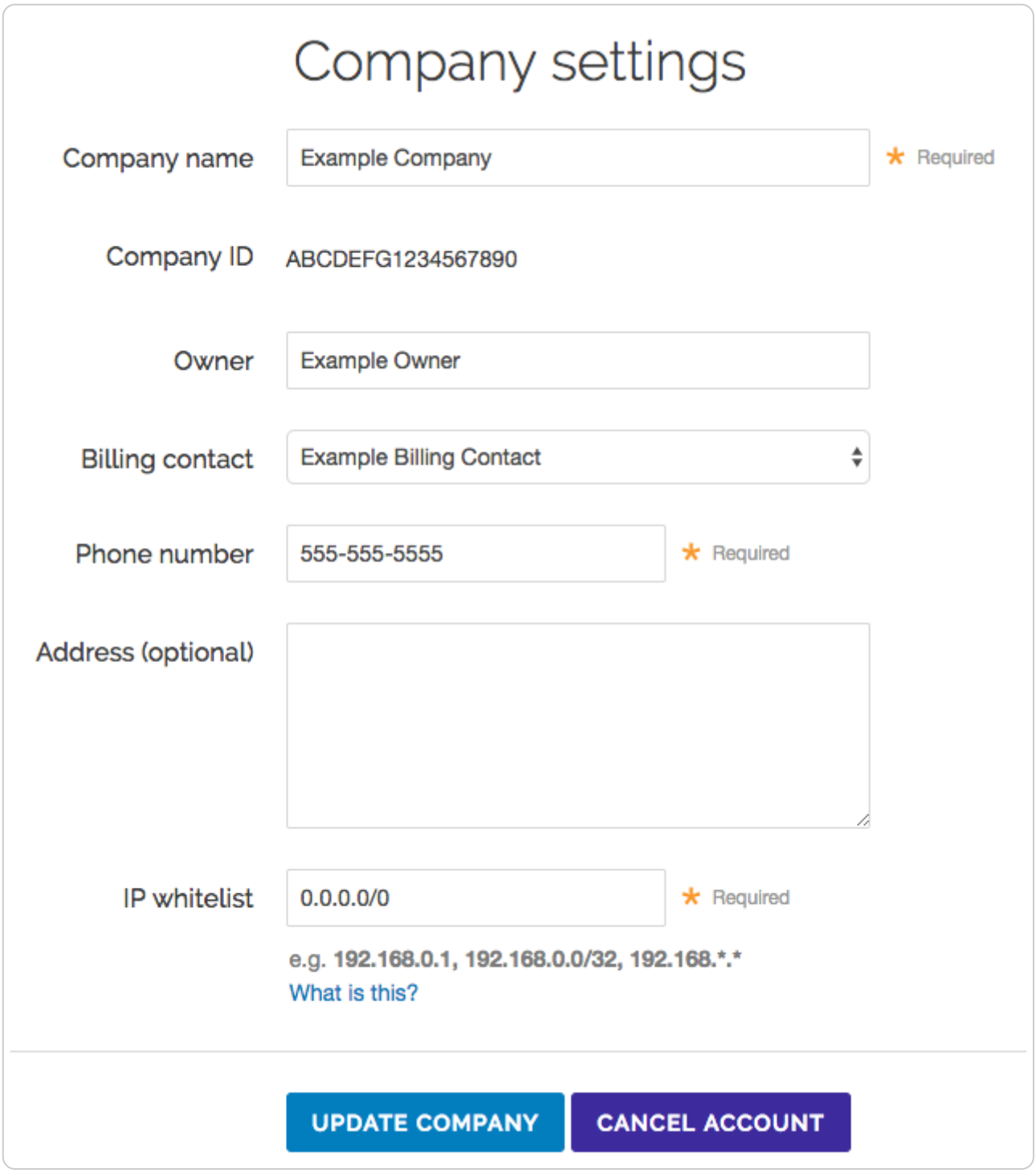
1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.
2. Click the **User management** link. The User management page appears.
3. In the **User Invitations** area, click the **Invite** button. The Invite a new user window appears.



4. In the **Email** field, type the email address of the user you'd like to invite to become a billing contact.

   ★ **TIP:** To send invoices to multiple people, we recommend setting up a group email address and setting that email address as your Billing Contact user.

5. From the **Choose their role** options, select **Billing**.
6. Click the **Invite** button to send an invitation to the email you specified.
7. Once the user has accepted the invitation, return to the account information in the web interface.
8. Click the **Company settings** link.
9. In the **Company settings** area, select the user's name from the **Billing contact** menu.



10. Click the **Update Company** button to set the billing contact.

For existing users

To send the billing invoice to a user who already has an account, follow these steps.

1. Log in to the Fastly web interface and click the **Account** link from the user menu (/guides/basic-concepts/about-the-web-interface-controls#about-the-user-menu). Your account information appears.

2. In the **Company Settings** area, select the user's name from the **Billing Contact** menu. Make sure the user name you select has the correct role (/guides/user-access-and-control/configuring-user-roles-and-permissions) assigned to view and manage billing information.
3. Click **Update Company** to save the billing contact.

---

§ Adding CNAME records (/guides/basic-setup/adding-cname-records)

This guide describes how to choose the right hostname and how to update the CNAME record for your domain with your DNS provider. Choosing the appropriate CNAME record (https://en.wikipedia.org/wiki/CNAME_record) is the final step required before Fastly can start acting as a reverse proxy (/guides/about-fastly-services/how-caching-and-cdns-work) and begin routing client traffic through Fastly services instead of directly to your origin.

Before you begin

Before you add a DNS CNAME record, keep in mind the following:

- To make the changes suggested here you must have access privileges to modify DNS records for your domain.
- If you plan to use Fastly on your apex domain (e.g., `example.com` rather than `www.example.com`), you can't use a CNAME record. See our guide to using Fastly with apex domains (/guides/basic-configuration/using-fastly-with-apex-domains) for more details.

Choosing the right Fastly hostname for your CNAME record

To successfully update your DNS CNAME record, you must choose the right Fastly hostname to use. The hostname you choose will differ based on:

- the standard HTTPS (TLS) support requirements for your domain, including whether or not HTTP/2 is enabled.
- any custom TLS options (/guides/detailed-product-descriptions/tls-service-options) purchased for your domain.

- whether or not you choose to limit your traffic (/guides/performance-tuning/enabling-global-pops#limiting-pop-use-to-north-america-and-the-european-union) to the North American and EU network or use Fastly's global network (/guides/performance-tuning/enabling-global-pops).

We've provided recommendations below based on these criteria.

Non-TLS hostnames and limiting traffic

If you don't require TLS support and only need to accept HTTP (Port 80) connections, use one of the following hostnames:

- Use `nonssl.global.fastly.net.` to route traffic through Fastly's entire global network.
- Use `nonssl.us-eu.fastly.net.` to route traffic through Fastly's North American and EU POPs only.

> ❗ **IMPORTANT:** Fastly's non-TLS hostnames refuse HTTPS connections (port 443) to prevent TLS certificate mismatch errors.

TLS-enabled hostnames

If you've purchased either a Shared TLS Certificate (/guides/detailed-product-descriptions/tls-service-options#shared-tls-certificate-service) or Shared TLS Wildcard Certificate (/guides/detailed-product-descriptions/tls-service-options#shared-tls-wildcard-certificate-service) service, use one of the following HTTP/1.x and HTTP/2 enabled hostnames:

- Use `[letter].shared.global.fastly.net.` to route traffic through Fastly's entire global network.
- Use `[letter].shared.us-eu.fastly.net.` to route traffic through Fastly's North American and EU POPs only.

When you purchase one of these certificate services, Fastly Support (mailto:support@fastly.com) will add your domains to a specific TLS Certificate, usually differentiated by a certificate letter (e.g., `a`, `a2`, `b`, `c`). You'll need to add the appropriate certificate letter to the beginning of the Fastly hostname noted above for use in your CNAME record. For example, if your domain was added to our `a` certificate and was being routed through Fastly's entire global network, the above hostname would become:

`a.shared.global.fastly.net.`

> ❗ **IMPORTANT:** You must use the assigned Fastly TLS hostname provided by Fastly Support. Using the incorrect Fastly hostname will cause a TLS Certificate mismatch error (/guides/debugging/tls-origin-configuration-messages) for HTTPS (Port 443) traffic.

Customer-specific hostnames

If you've purchased our Hosting Service for Customer-Provided TLS Certificates (/guides/detailed-product-descriptions/tls-service-options#hosting-service-for-customer-provided-tls-certificate) or our Customer-Provided TLS Certificate Hosting Service (/guides/detailed-product-descriptions/tls-service-options#customer-provided-tls-certificate-hosting-service) option, we'll assign you to a specific domain map that uses the following format:

`[name].map.fastly.net.`

Free TLS wildcard Certificate

If you plan to accept both HTTP (port 80) and HTTPS (port 443) connections and you're using Fastly's free shared TLS wildcard certificate (/guides/securing-communications/setting-up-free-tls), use:

`[name].global.ssl.fastly.net.`

IMPORTANT: The free TLS hostname does not support use with your own domain name (`www.example.com`). Customers typically use this the free TLS hostname in links directly to assets (e.g., linking to `https://example.global.ssl.fastly.net/example.jpg`) or for testing purposes. If you want to use your own domain (`www.example.com`), see the TLS-enabled hostname section above.

Updating the CNAME record with your DNS provider

Once you've determined the appropriate Fastly hostname for your domain, the next step is to create a CNAME record for your domain. The steps you follow will vary depending on your DNS provider's control panel interfaces. Refer to your DNS provider's documentation for exact instructions on how to create or update a CNAME record.

TIP: If you can't find your provider's CNAME configuration instructions, Google maintains instructions for most major providers (https://support.google.com/a/topic/1615038?hl=en). Keep in mind that these instructions are maintained by Google, not Fastly, and are tailored specifically for Google enterprise services.

If you run your own DNS server or are familiar with the format of BIND zone files, the CNAME record would look similar to this:

```
www.example.com.    3600    IN    CNAME    nonssl.global.fastly.net.
```

In the above example, the domain set up on Fastly is `www.example.com.`, with a time-to-live (TTL) of `3600` seconds (1 hour), the Record Type is `CNAME`, and the Fastly hostname is `nonssl.global.fastly.net.` because TLS support isn't required and traffic will be routed through Fastly's entire global network.

Best practices when updating a DNS CNAME record

- Be sure you've added all domains you want served by Fastly to the appropriate service. If you don't and you point your domain to Fastly, an `unknown domain` error will occur.
- Make sure your service is properly configured. You can test a Fastly service on your local machine by using cURL (/guides/debugging/curl-and-other-caching-verification-methods) and our Testing setup before changing domains (/guides/basic-configuration/testing-setup-before-changing-domains) guides.
- If you have multiple hostnames on the same domain (e.g., `api.example.com`, `www.example.com`, `app.example.com`), you can use a DNS wildcard record (`*.example.com`) at your DNS provider so only a single CNAME record is created and maintained. You should also add either a matching `*.example.com` domain or the individual domains to your Fastly service.
- Before changing a CNAME to point to a Fastly hostname, change your service configuration to lower the CNAME's TTL to a small number (we suggest 60 seconds) and wait for the old TTL to expire. Creating a DNS CNAME record for your domain after the TTL expiration ensures you have an easy way to roll back changes if you encounter an issue. Once you confirm everything is working properly using Fastly, you can increase the TTL to its original value.

Checking your CNAME record

To check your CNAME record, run the following command in a terminal window:

`dig www.example.com +short`

Your output should appear similar to the following:

```
nonssl.global.fastly.net.
151.101.117.57
```

In most cases, the hostname displayed first will be your current Fastly hostname (in this case, `nonssl.global.fastly.net.`). If you don't see a Fastly hostname in the output or if you see an incorrect Fastly hostname, then either your CNAME isn't properly set at your DNS provider or an older CNAME record is still cached by your local DNS resolver.

You can use various online DNS query tools like OpenDNS Cache Check (https://cachecheck.opendns.com/) or whatsmydns.net (https://www.whatsmydns.net/) to test the current DNS responses from the different DNS resolvers worldwide.

Removing CNAME records

If you deactivate a service (/guides/basic-setup/working-with-services#deactivating-a-service), delete a service (/guides/basic-setup/working-with-services#deleting-a-service), or cancel your account (/guides/account-types-and-billing/accounts-and-pricing-plans#canceling-your-account), we strongly recommend modifying or deleting any CNAME records pointing to Fastly hostnames. Follow the instructions on your DNS provider's website. Doing so will minimize the risk of unauthorized use of your domains.

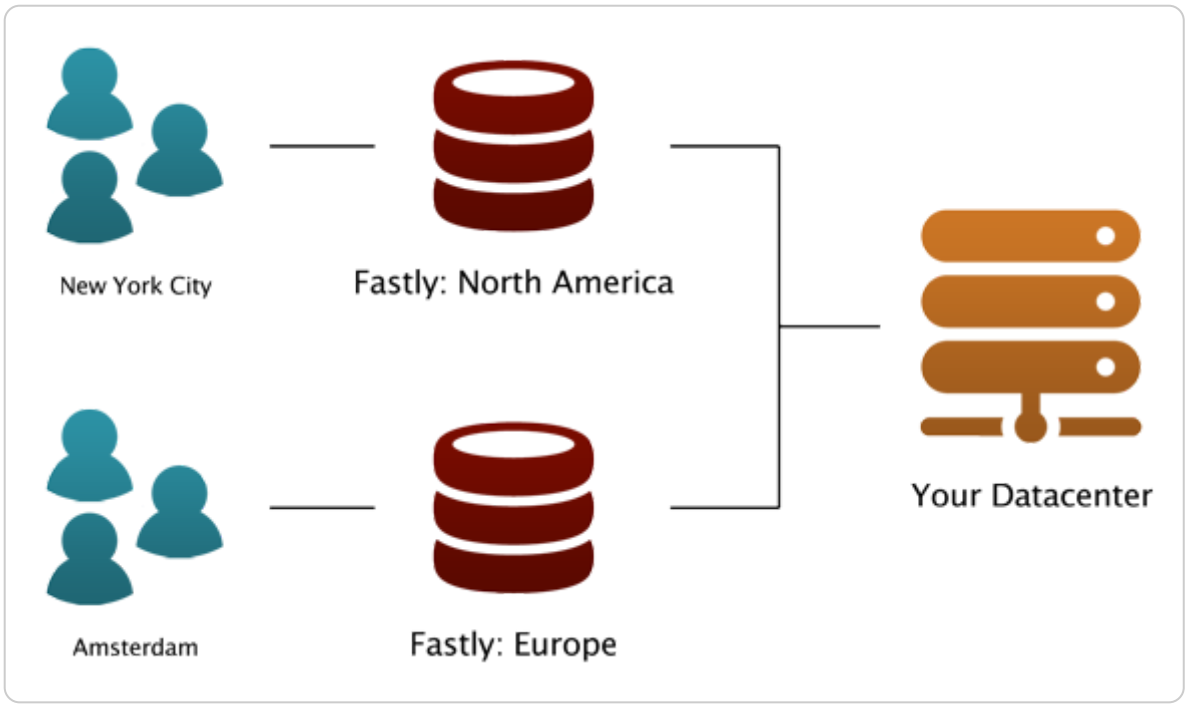
---

§ Getting started with Fastly (/guides/basic-setup/getting-started-with-fastly)

In this article, we explain what Fastly does and how best to use it with your site.

How Fastly works

Fastly works by storing the content of your website on servers all over the world and quickly delivering that content to your users. We do this using Varnish (https://www.varnish-cache.org), an open source web application accelerator.



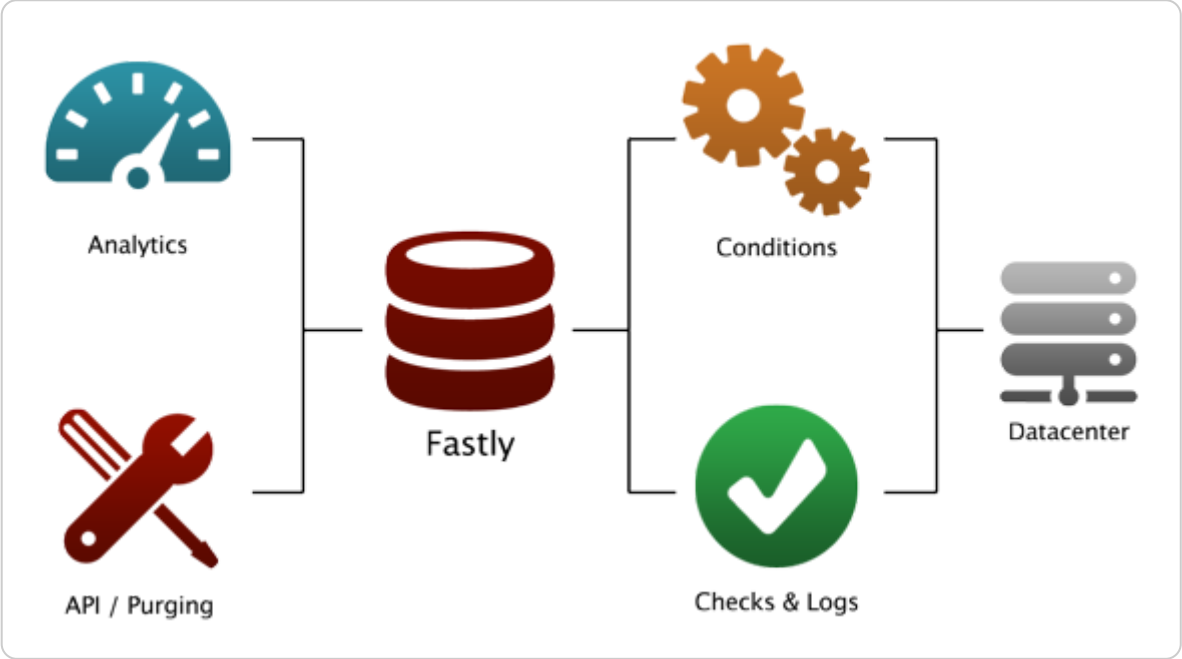
We track the geo-location of each user and make sure they are connecting to a server that is closest to them. This makes your site faster by reducing the time spent waiting for data to be sent from the server to the user.

We also give you full control over when and how we store content from your servers. You can set a Time To Live (TTL) for any path on your site and instantly invalidate or purge any path on your site using our Purge API (/api/purge).

By using these tools, you only have to generate pages one time for the site for many millions of page views. This saves time for your users and costs on your server bills.

Advanced features

Fastly also provides many advanced features that help you monitor how your data is accessed and customize your content delivery.



- **Instant Purging (/guides/purging/)** allows you to have better control over when and how content is updated. You can update your data when you want and as often as you want, rather than waiting up to 24 hours to change data at the edge.
- **Real Time Analytics (/guides/basic-concepts/about-the-web-interface-controls)** provides a top level view of your network and how your site is performing. Every second, we compile the relevant data about all of your traffic into an easy-to-read report.
- **Conditions (/guides/conditions/)** change how requests are routed, what headers to send, and how content is cached.
- **Health Checks (/guides/basic-configuration/health-checks-tutorial)** monitors the status of one or many of your back end servers. This way if anything goes wrong with your servers, you immediately know about it.

- **Streaming Logs (/guides/streaming-logs/setting-up-remote-log-streaming)** are quickly and easily configured to send information from your servers anywhere and in the format you want.
- **Varnish Configuration Language (VCL) (/guides/vcl/guide-to-vcl)** allows you to modify nearly every aspect of an HTTP request and response. You can upload VCL files with specialized configurations to your account.
- The **Fastly API (/api/)** can programmatically handle your configuration. This allows you to write scripts to handle basic configuration tasks and create your own administrative views (so they can be directly coupled with your existing admin software).

Getting started

If you do not have an account, sign up now (https://www.fastly.com/signup). Feel free to choose the developer plan so you can test how Fastly works on your site.

If you want to use Fastly but do not know where to begin, check the Basic setup (/guides/basic-setup/) documentation. You can learn everything you need to set up and configure your first service for your site.

If you want to explore more, check the Basic configuration (/guides/basic-configuration/) documentation to learn more about caching, and features such as purging (/guides/purging/) and shielding (/guides/performance-tuning/shielding).

If you want information on advanced features, especially related to things like load balancing (/guides/performance-tuning/load-balancing-configuration) or the Varnish Configuration Language (/guides/vcl/) that we support, check the advanced configuration section of our help files or the API Reference (/api/), which includes a full reference to the Fastly API.

If you are having problems, send us a message at support@fastly.com (mailto:support@fastly.com).

---

§ Glossary of terms (/guides/basic-setup/glossary-of-terms)

These are common Fastly, HTTP, and networking terms you may encounter within our service guides.

ACL

Access Control List. A list of permissions that can be attached to an object (/guides/vcl/using-access-control-lists) allowing customers to quickly check a client's IP against a list of known net blocks and then make decisions based on the result.

Altitude

Fastly's customer summit (https://www.fastly.com/altitude).

Backend

See *origin server*.

Cache-Control

The specific HTTP header (/guides/tutorials/cache-control-tutorial) that controls who can cache a response, under which conditions, and for how long. Fastly respects `Cache-Control` headers returned from origin servers as one approach to cache management. See also *surrogate-control*, *max-age*, and *s-maxage*.

community.fastly.com

Fastly's community discussion forum (https://community.fastly.com/).

Cookie

HTTP headers used to perform certain functions like authenticating login in secure website areas, information tracking, remembering user preferences, and customizing how information is presented.

cURL

An open-source command line tool (https://curl.haxx.se/) for transferring data with URL syntax from or to a server using one of many supported protocols. Fastly users can issue cURL commands to verify requests are caching (/guides/debugging/curl-and-other-caching-verification-methods) in the Fastly network.

DNS

Domain Name System. A system for naming computers and network services that translates a domain's numbered IP address into an easy-to-remember alphabetic name.

Edge Dictionary

A type of container Fastly users can create (/guides/edge-dictionaries/) to store data as key-value pairs and turn frequently repeated statements into a single function that acts as constant.

Egress traffic

Bandwidth used when traffic travels from Fastly *points of presence* (POPs) to the end user.

ESI

Edge Side Includes. An XML-based markup language (http://www.w3.org/TR/esi-lang) that allows content assembly by HTTP surrogates. Allows Fastly users to cache pages that contain both cacheable and uncacheable content (such as user-specific information).

Gzip

A way of compressing information to make it faster to transmit. Fastly allows users to dynamically gzip content (/guides/basic-configuration/enabling-automatic-gzipping) based on file extension or content type.

Header

An HTTP field that precedes the main content of information being sent in a request or response and describes the length of the content, type of content, or other characteristics of the information.

Host (header)

Information used in addition to the IP address and port number to uniquely identify a domain.

Ingress traffic

Bandwidth used when end users make requests that send traffic to Fastly *points of presence* (POPs).

Instant Purge

A feature of Fastly's purging functionality (/guides/purging/) that allows users to actively invalidate content in Fastly caches within milliseconds. See also *Soft Purge*.

manage.fastly.com

The web interface through which customers access Fastly's CDN services (https://manage.fastly.com/).

max-age

An HTTP *Cache-Control* directive that specifies how long (in seconds) an object will remain in the cache before Fastly removes the object from storage. See also *surrogate-control*, *cache-control*, and *s-maxage*.

MTR

A tool that combines traceroute and ping programs in a single network diagnostic tool. Frequently used in debugging network connections (/guides/debugging/debugging-with-mtr).

Origin server

The location or address from which Fastly's network requests the content it will serve.

Origin Shield (Shielding)

A specific Fastly *point of presence* (POP) designated by users (/guides/performance-tuning/shielding#enabling-shielding) as the primary source of content through which all content requests from other POPs will be directed in lieu of contacting a customer's origins directly.

OTFP

On-the-fly packaging. A feature of Fastly's video on demand (https://www.fastly.com/services/video-demand) media and streaming offering that allows customers to dynamically package video (/guides/detailed-product-descriptions/about-fastlys-onthefly-packaging-service) for delivery in multiple HTTP streaming formats. Also known as "just in time" video content packaging.

POP

Point of Presence. Datacenter within which Fastly's globally distributed cache servers (https://www.fastly.com/network-map) reside.

Priority

A setting that allows users to specify the order request and cache settings execute within their subroutines. The Priority can be any whole number and always default to 10. The smaller the assigned priority number, the sooner that condition executes (e.g., 1 executes sooner than 10).

private

An HTTP *Cache-Control* directive that allows users to select which objects are not cached. Fastly will not cache responses with a Cache-Control value of `private`.

Purging

The process of picking out one or more objects from the Fastly cache and discarding it along with its variants. See also *Instant Purge* and *Soft Purge*.

Redirect

A function that directs requests for information from their originally intended locations to a more desirable destination (/guides/performance-tuning/generating-http-redirects-at-the-edge).

Role-based access control

A method of regulating access to resources based on the roles (/guides/user-access-and-control/configuring-user-roles-and-permissions) of the individuals within an organization.

s-maxage

An HTTP cache control directive similar to *max-age*, but applied only to shared caches. See also *surrogate-control* and *cache-control*.

Service

A user-defined set of caching rules and behavior for a website or application. You can use the Fastly web interface to create, edit, and delete your services (/guides/basic-setup/working-with-services).

Service authorization

A function that grants per service access to an Engineer (/guides/user-access-and-control/configuring-user-roles-and-permissions#user-roles-and-what-they-can-do) role. In addition, the API calls (/api/account#service_authorizations) that limit user access to specified

services. See also *service permissions*.

Service permissions

The functions that manage the level of control (/guides/user-access-and-control/configuring-user-roles-and-permissions#service-access-and-permission-levels) granted to an Engineer (/guides/user-access-and-control/configuring-user-roles-and-permissions#user-roles-and-what-they-can-do) role once they've been authorized to access a service. See also *service authorization*.

Set-Cookie

The header sent by a server in response to an HTTP request and then used to create a cookie on a user's origin. Fastly supports a method for extracting a named value (/guides/vcl/response-cookie-handling) out of `Set-Cookie` headers no matter how many there are. By default, Fastly will not cache responses that contain a `Set-Cookie` header.

Soft Purge

A type of purging (/guides/purging/) that allows users to easily mark content as outdated (/guides/purging/soft-purges) (expired) instead of immediately deleting it from Fastly's caches. See also *Instant Purge*.

status.fastly.com

Fastly's network status (https://status.fastly.com/) monitoring site. Allows customers to quickly check whether anomalies they see may be due to a known problem currently being worked on by Fastly or if their issues more likely stem from problems within their own infrastructure.

support@fastly.com

The main email address of Fastly's Customer Support (https://www.fastly.com/support) team through which customers can ask questions and receive assistance.

Surrogate-Control

An HTTP response header that allows origin servers to use control directives to dictate how intermediate caches, including Fastly, should handle response entities. `Surrogate-Control` will not affect browsers. See also *cache-control*, *max-age*, and *s-maxage*.

Surrogate Key

A unique identifier that allows customers to group content together for faster processing. Fastly uses surrogate keys (/guides/purging/getting-started-with-surrogate-keys) as part of its purging strategy (/guides/purging/).

Synthetic response

Custom responses generated within the CDN that users can set if a specific URL is requested or a specific condition, such as a status code, is met. These responses require no origin server interaction.

TLS (SSL)

A cryptographic protocol Fastly follows (/guides/securing-communications/) that ensures privacy between communicating applications and their users on the internet.

URL

Uniform Resource Locator. An address used (/guides/basic-configuration/working-with-domains) to find a site or application's objects on the internet.

Varnish

Caching software that helps content-heavy dynamic websites as well as heavily consumed APIs load faster. Fastly's core caching infrastructure is based on a heavily modified version of Varnish (/guides/vcl/guide-to-vcl#about-varnish-and-why-fastly-uses-it).

VCL

Varnish Configuration Language. A scripting language (/guides/vcl/guide-to-vcl) used to configure and add logic to Varnish caches. Fastly users can create custom VCL files with specialized configurations.

WAF

Web Application Firewall (/guides/web-application-firewall/web-application-firewall). A network security system that monitors, filters, or blocks data packets as they travel to and from a web application.

www.fastly-debug.com

A network debugging tool designed to provide key info to help a Fastly user troubleshoot issues with Fastly's Customer Support (https://www.fastly.com/support) team.

---

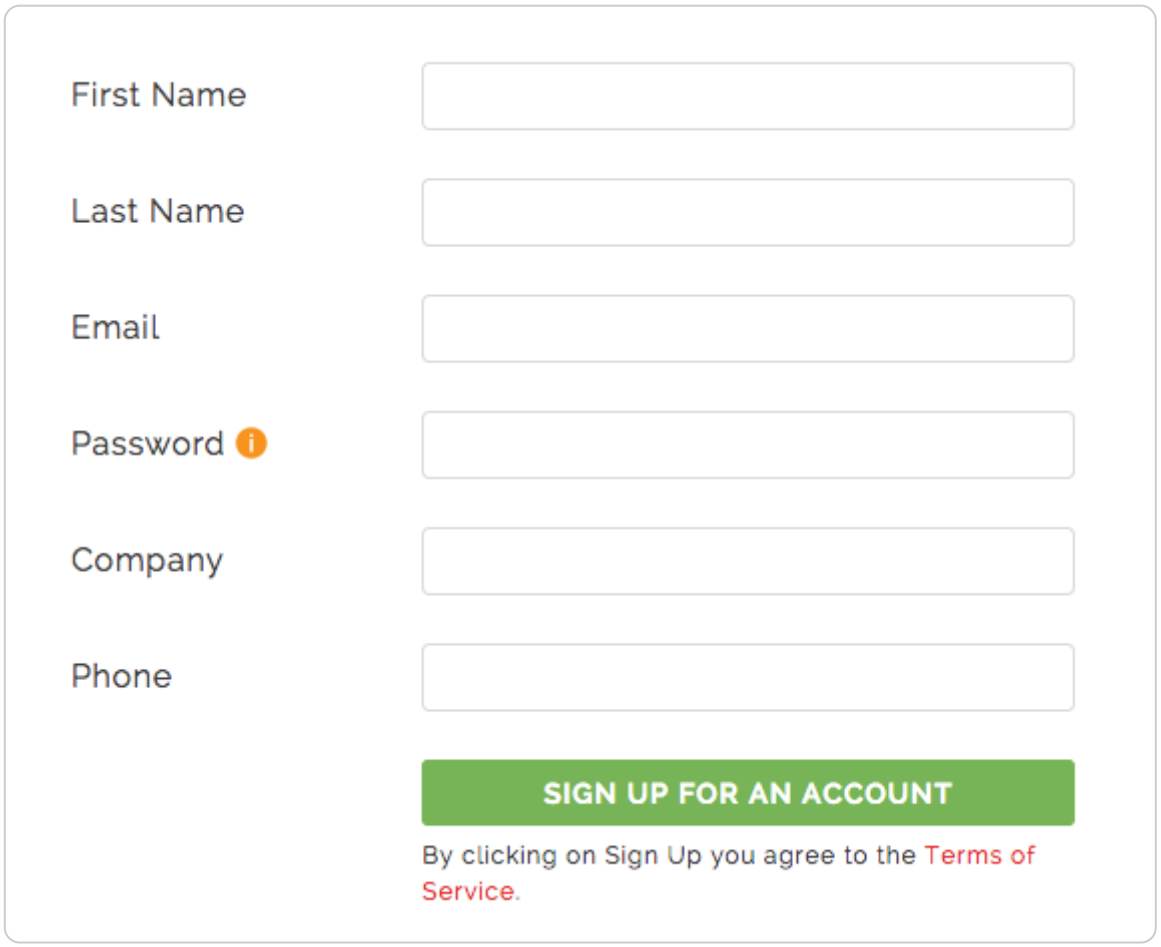
§ Sign up and create your first service (/guides/basic-setup/sign-up-and-create-your-first-service)

To create a Fastly account and set up your first service, follow the steps below.

Sign up at Fastly.com

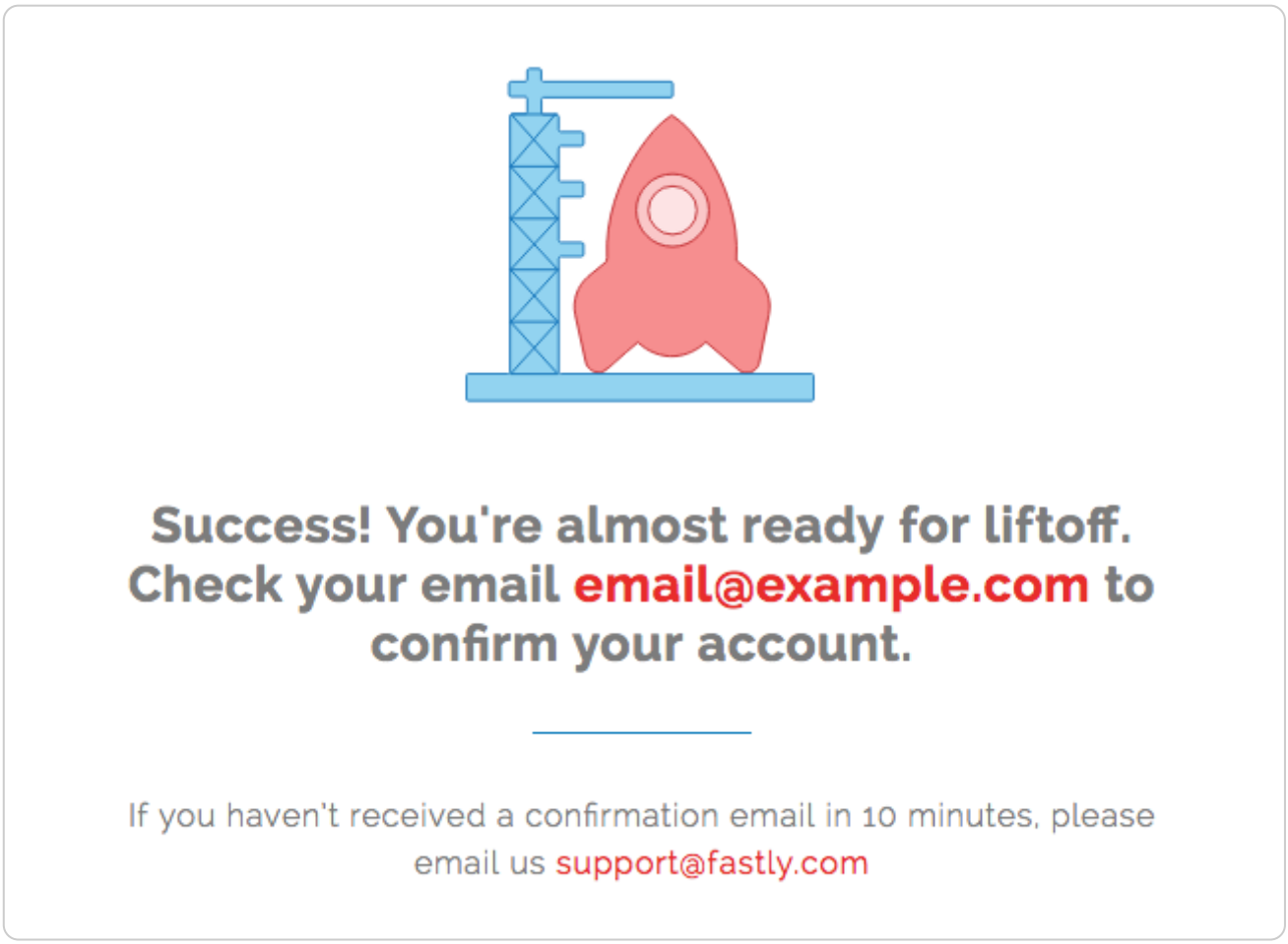
Before you do anything else, you must sign up for a Fastly account.

1. Click on any **Sign Up** button on the Fastly.com website or simply point a browser to the signup form (https://www.fastly.com/signup).
2. When the signup form appears, fill in all the fields with your contact information. All the fields are required.



NOTE: You'll be able to change your password and email address (/guides/user-access-and-control/email-and-password-changes) any time after signup. Without a valid email we can't send you account verification details during account setup. Without a valid telephone number, we can't assist you with specific kinds of account lockout issues (/guides/account-management-and-security/enabling-an-ip-whitelist-for-account-logins).

3. Click the **Sign Up For An Account** button. The confirmation screen will appear with instructions on what to do next and you'll be sent an e-mail that contains a verification link.

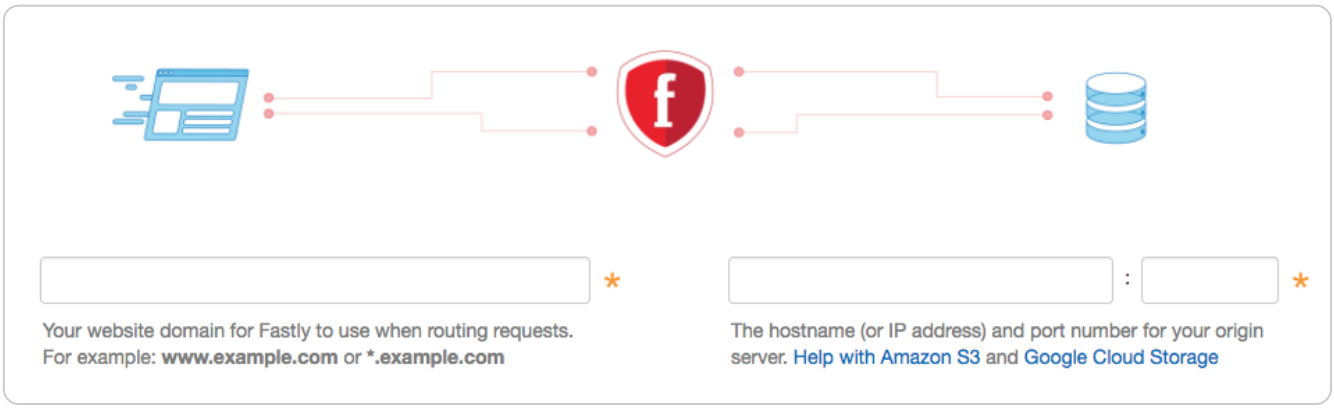


4. Check your inbox and find the confirmation email we sent you.
5. Click the verification link (we need to make sure you're not a spam robot and verify your email). The verification link will immediately take you to the first step of the quick start process so you can create your first service.

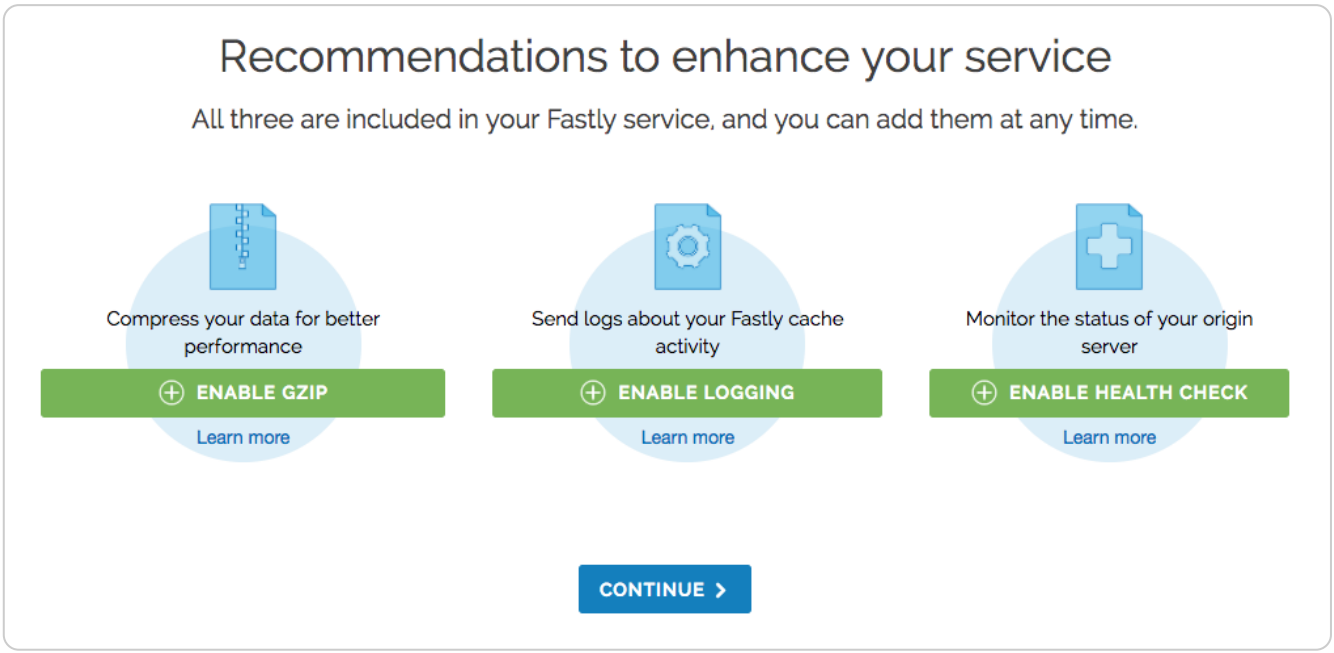
Create your first service

Once you've verified your email, we log you into the application automatically and immediately take you through the quick start process to create your first service.

1. Specify your **domain** and **hostname**. We need this information to properly route requests to your website and to make sure your cache updates properly. If you have many servers you can configure the rest later.



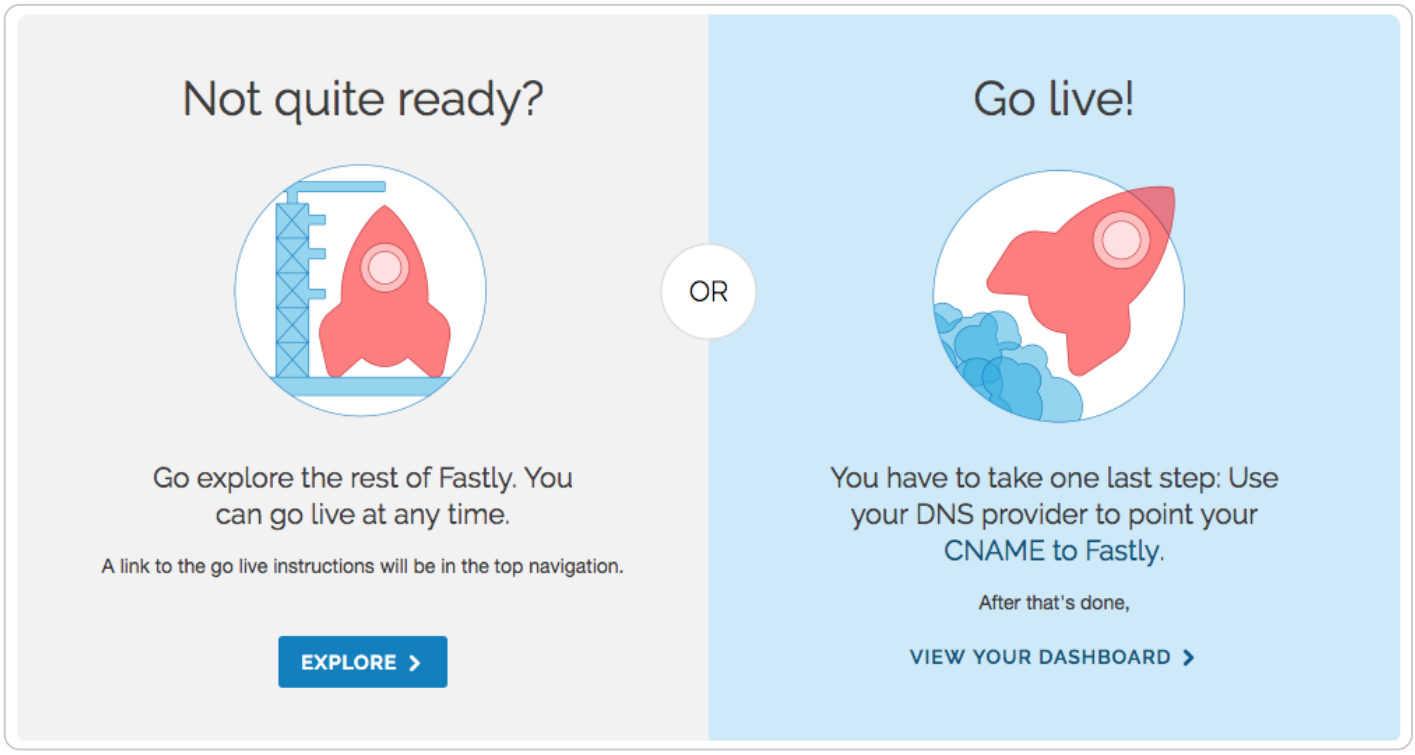
2. Click the **Continue** button.



3. Optionally enable the following features:
   - Gzip (/guides/basic-configuration/enabling-automatic-gzipping) to dynamically gzip content, resulting in smaller file sizes and faster transfer speeds.
   - Logging (/guides/streaming-logs/setting-up-remote-log-streaming) to automatically store logs with a third-party service, resulting in the ability to view cache activity and to troubleshoot.
   - Health checks (/guides/basic-configuration/health-checks-tutorial) to monitor the health of your origin server so that Fastly will stop attempting to send requests to it when it is unhealthy.
4. Click the **Continue** button. The system configures your service.
5. To test whether or not the configuration was successful, open **yourdomain.global.prod.fastly.net** in a new browser window (replace `yourdomain` with your own website's domain name). Your website should appear, but it may take up to 60 seconds for the new configuration settings to take effect.

Add CNAME DNS records

You must set the CNAME DNS record for your domain to point to Fastly. For more information, see the instructions in our Adding CNAME records (/guides/basic-setup/adding-cname-records) guide.



Once you've completed this step, you should be all ready to go!

If you have any problems, feel free to contact us at support@fastly.com (mailto:support@fastly.com).

§ Working with services (/guides/basic-setup/working-with-services)

A service is a user-defined set of caching rules and behavior for a website or application. The Fastly web interface allows you to create new services or edit existing ones and then activate new versions of them that include your changes once you have things configured the way you want. The web interface also allows you to do other things with existing services, like rename them, compare them to each other, deactivate or reactivate them, and delete them.

Creating and editing services

You can create a new service or edit an existing one depending on how you manage your web properties and the digital assets that make them up. You might create a new service when you want to do things like:

- add a new website you control to your list of web properties
- add a new domain to your growing list of existing domains already served by Fastly
- isolate traffic metrics for specific digital assets, like a site's images

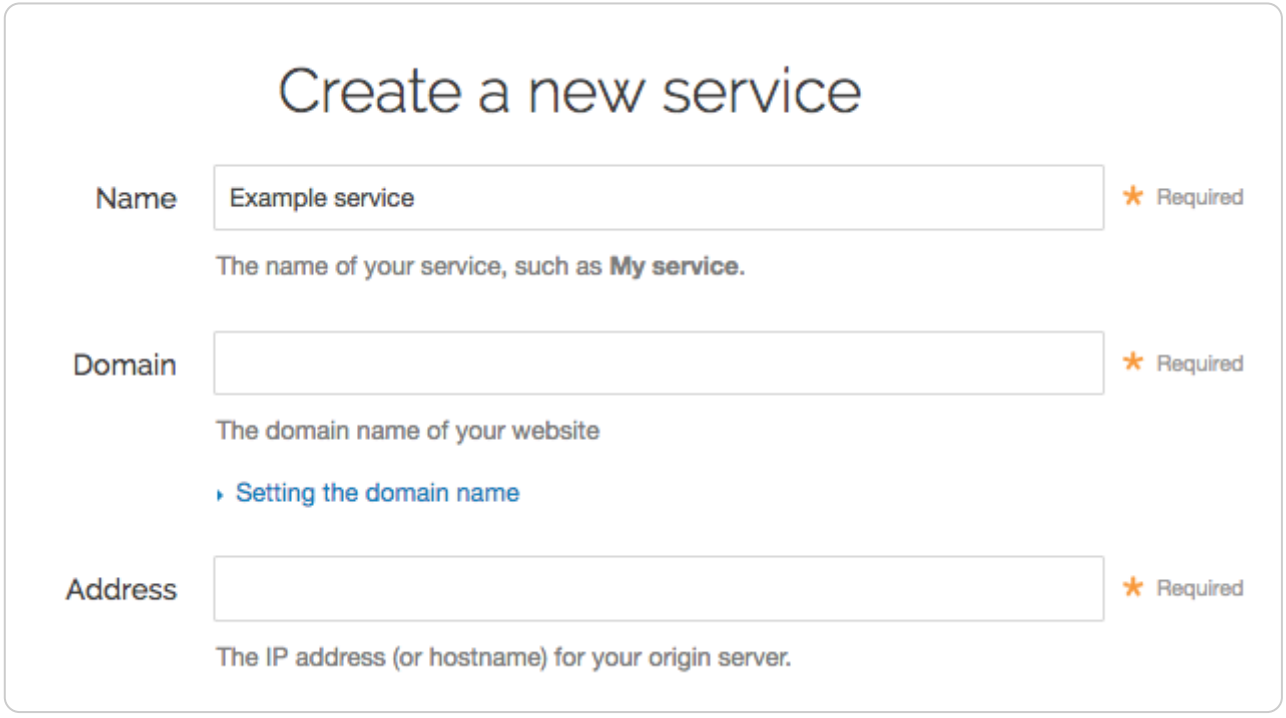
You might edit a version of an existing service when you want to do things like:

- change the amount of time information is retained in cache memory for a service
- configure a service to temporarily serve stale content should your origin server need to be unavailable for an extended period of time (for example, taken offline for maintenance)
- decrease the amount of time Fastly will wait for your origin server to respond to a request for content

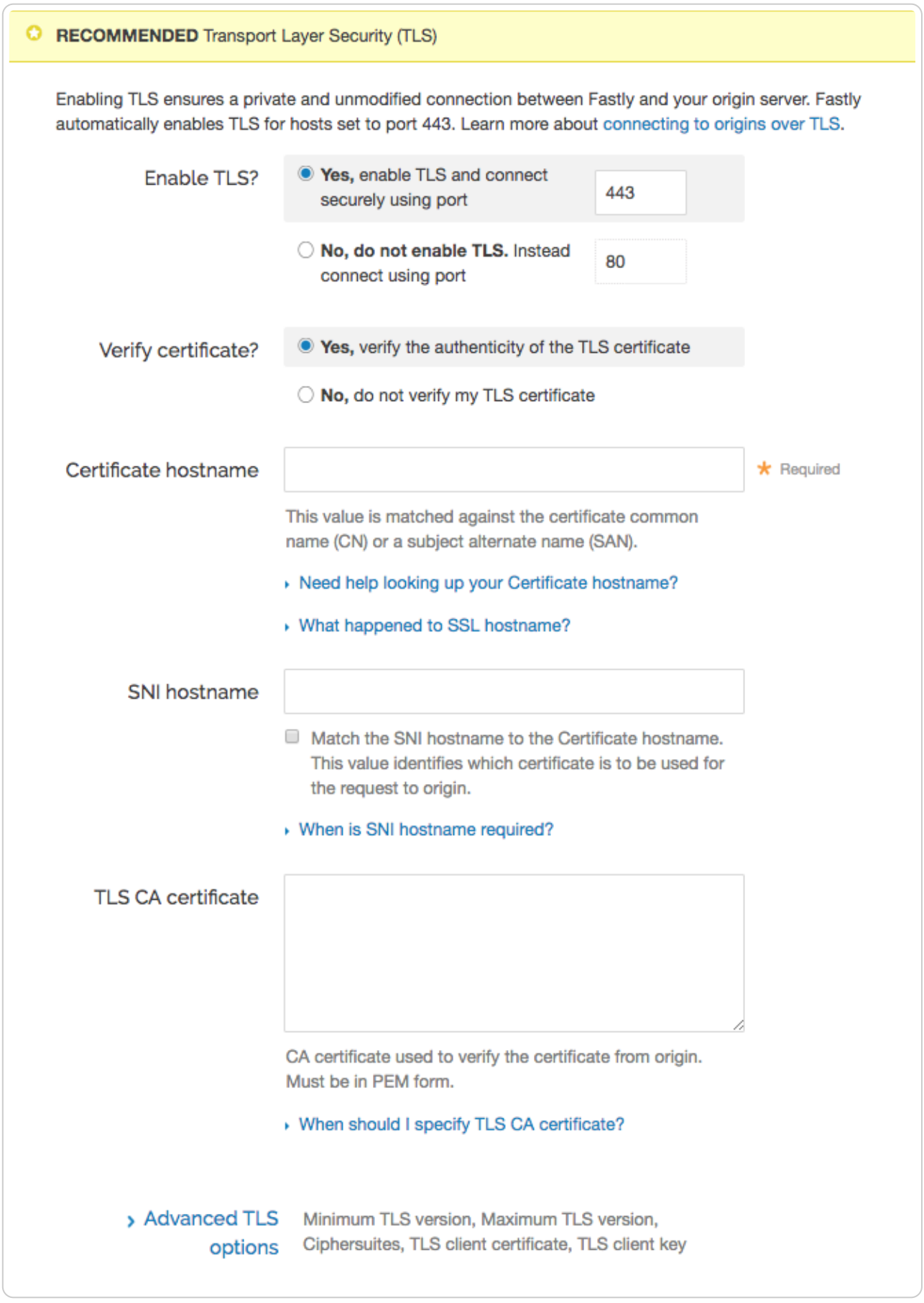
Creating a new service

To create a new service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. Click the **Create service** button. The Create a new service page appears.



3. Fill out the **Create a new service** page as follows:
   - In the **Name** field, type a human-readable name for the new service. You can change this name at any time.
   - In the **Domain** field, type the domain name of your website. We need this information to properly route requests to your website.
   - In the **Address** field, type the IP address (or hostname) for your website's origin server.



4. Fill out the **Transport Layer Security (TLS)** area as follows:

- Leave the **Enable TLS?** default set to **Yes** if you want to enable TLS and secure the connection between Fastly and your origin. To enable TLS, a valid SSL certificate must be installed on your origin server and port 443 (or the specified port) must be open in the firewall. You can select **No** if you do not want to use TLS.
- Leave the **Verify certificate?** default set to **Yes** if you want to verify the authenticity of the TLS certificate. Selecting **No** means the certificate will not be verified.

  > ⚠ **WARNING:** Not verifying the certificate has serious security implications, including vulnerability to man-in-the-middle attack. Consider uploading a CA certificate instead of disabling certificate validation.

- In the **Certificate hostname** field, type your certificate hostname associated with your TLS certificate. This value is matched against the certificate common name (CN) (/guides/basic-configuration/connecting-to-origins#understanding-the-difference-between-certificate-hostname-and-sni-hostname-values) or a subject alternate name (SAN) depending on the certificate you were issued.
- In the **SNI hostname** field, optionally specify your SNI hostname. This is generally only required when your origin is using shared hosting, such as Amazon S3, or when you use multiple certificates at your origin. See Setting the TLS hostname (/guides/basic-configuration/connecting-to-origins#setting-the-tls-hostname) for more information.
- In the **TLS CA certificate** field, optionally include your TLS CA certificate. You may want to provide the CA certificate if you're using a certificate that is either self-signed or signed by a Certificate Authority (CA) not commonly recognized by major browsers. See Specifying a TLS CA certificate (/guides/basic-configuration/connecting-to-origins#specifying-a-tls-ca-certificate) for more information.

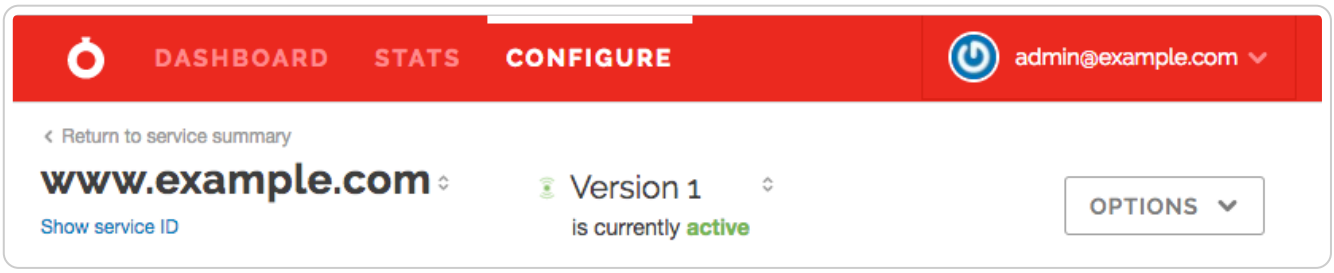
5. Click the **Create** button. The new service appears in the list of services available.

Editing and activating versions of services

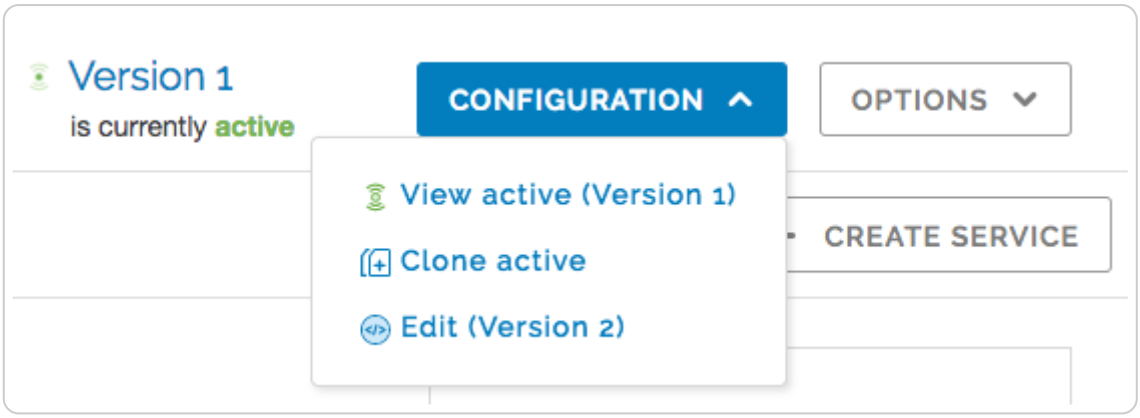
Fastly locks versions of services you've already activated to make rollbacks safer and provide version control. You can duplicate ("clone") any existing service version, active or inactive, and edit that cloned version. You must "activate" new versions of services in order to deploy their configurations. Configuration changes are never automatically activated.

To make changes to a service and activate a new version, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link. The Configure page appears.



2. Click the **Configuration** button. You can select **Clone active** to clone the active version of the service for editing, or you can select **Edit** to edit the latest draft of the service.



The service version page appears, listing the new version .



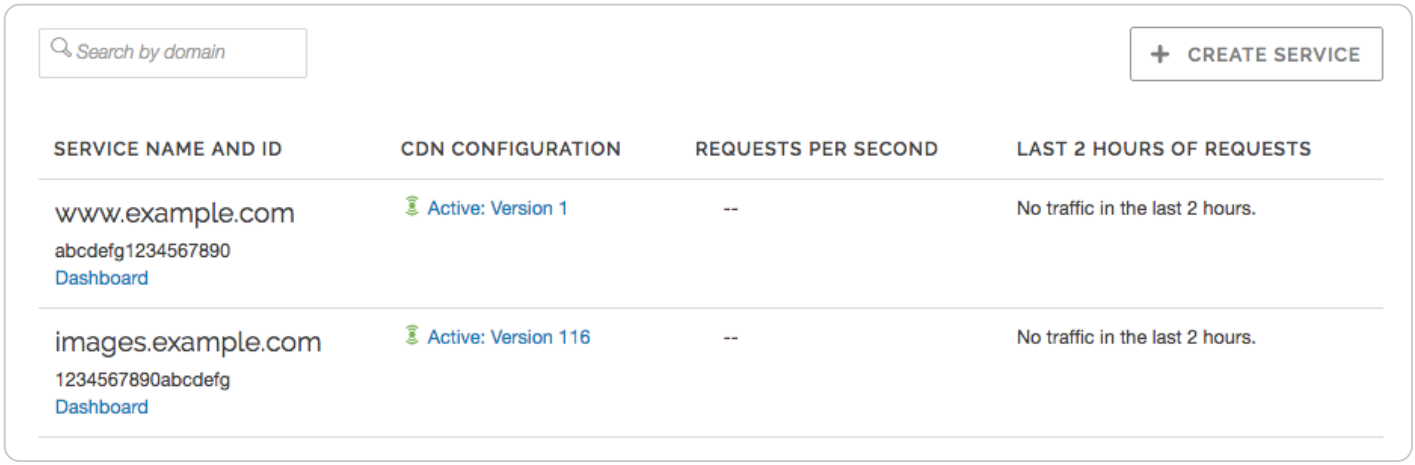
3. Click **Activate**. The new version of the service is activated and appears in the event log (/guides/monitoring-and-testing/monitoring-account-activity-with-event-logs).

Other things you can do

In addition to creating or editing services, you can view all your services, rename them, compare versions of them, deactivate or reactivate specific versions of them, and delete them.

Viewing all services

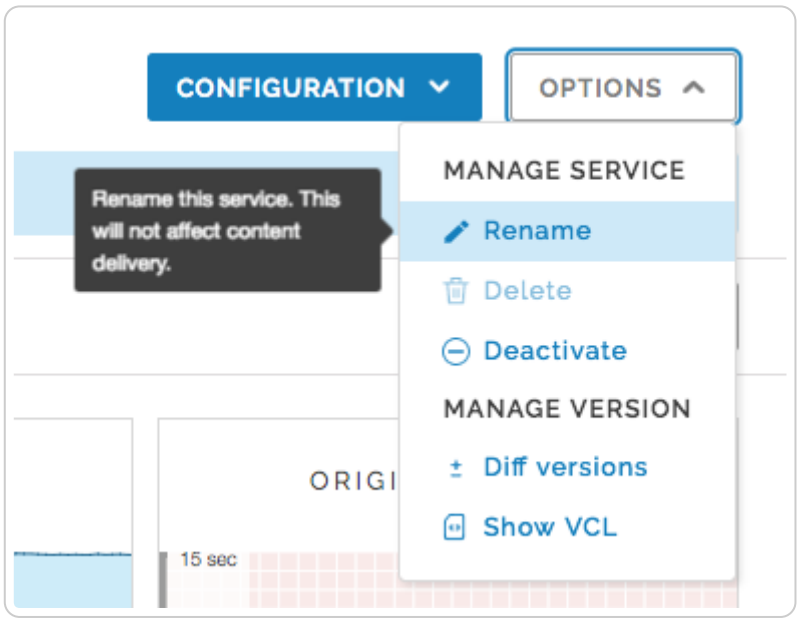
To view all your services, log in to the Fastly web interface. The All services page (/guides/basic-concepts/about-the-web-interface-controls#about-the-all-services-page) appears displaying a summary of all your services, sorted by requests per second.



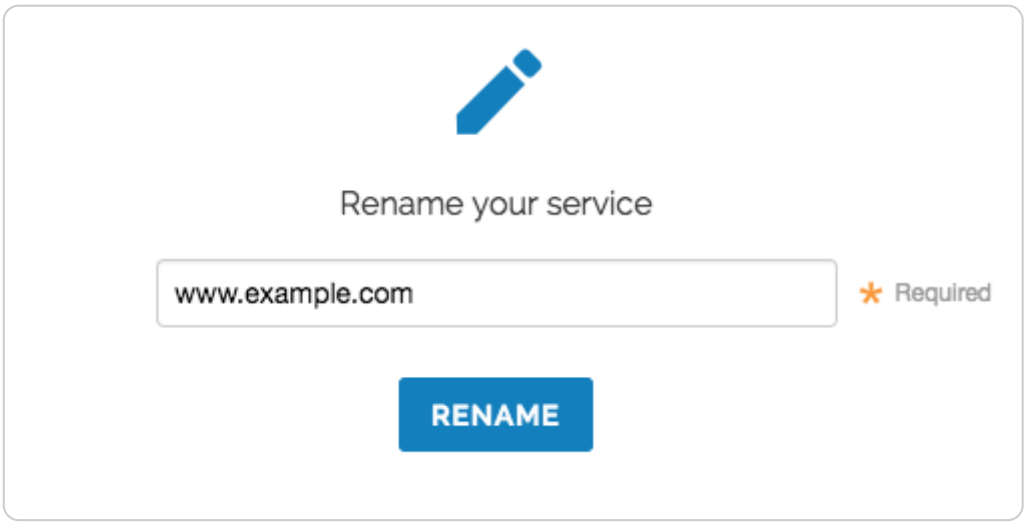
Renaming services

To rename your service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Options** button to open the **Manage service** menu and select **Rename**.



The Rename your service pane appears.

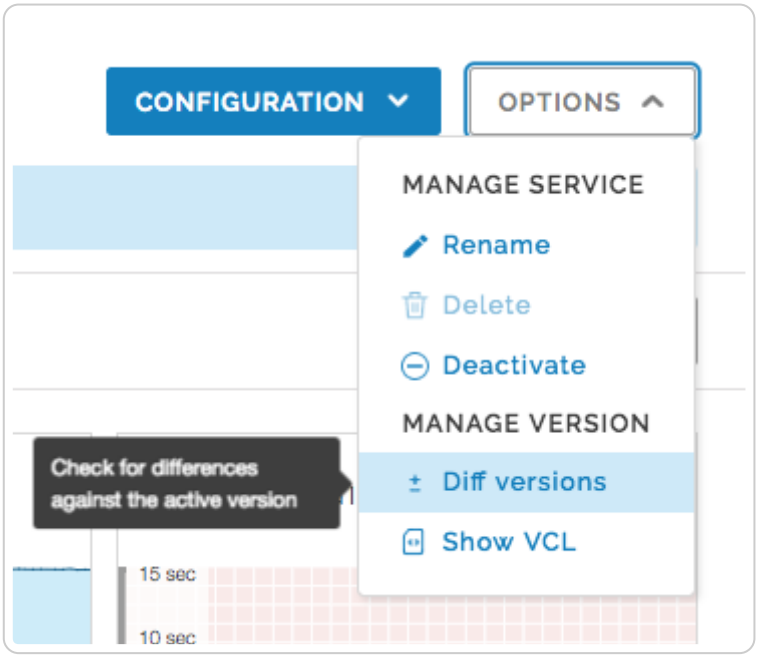


4. Type the new service name and then click **Rename**. The new service name appears.

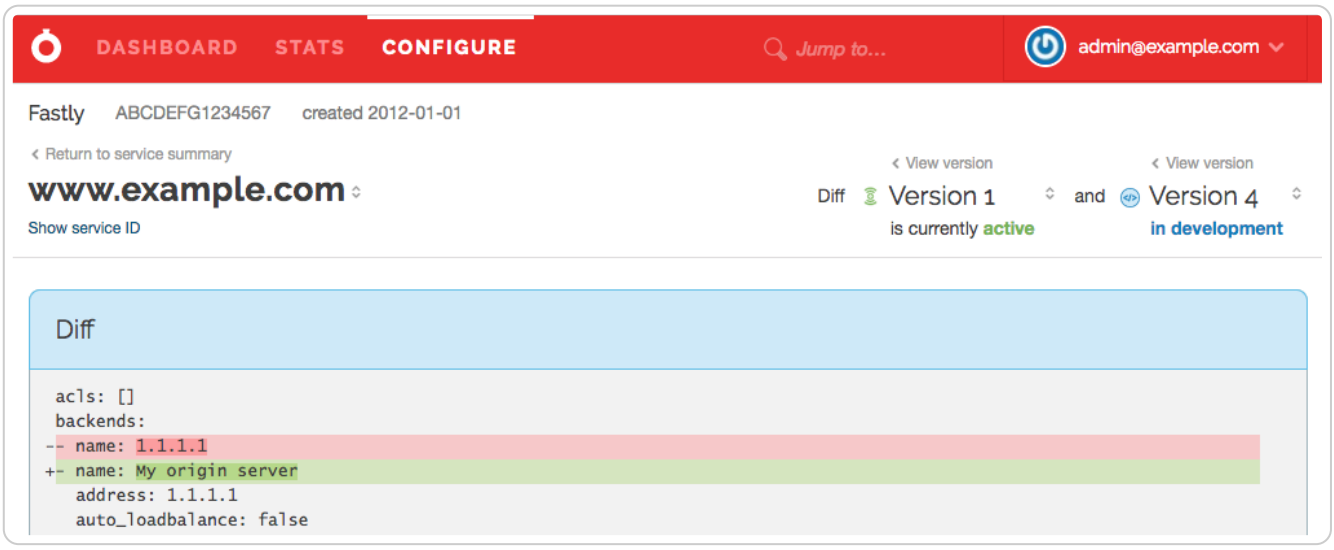
Comparing different service versions

To compare two versions of a service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Options** button to open the **Manage service** menu and select **Diff versions**.



The Diff versions page appears. Removals are highlighted in red with a minus sign at the beginning of the line. Additions are highlighted in green with a plus sign at the beginning of the line.

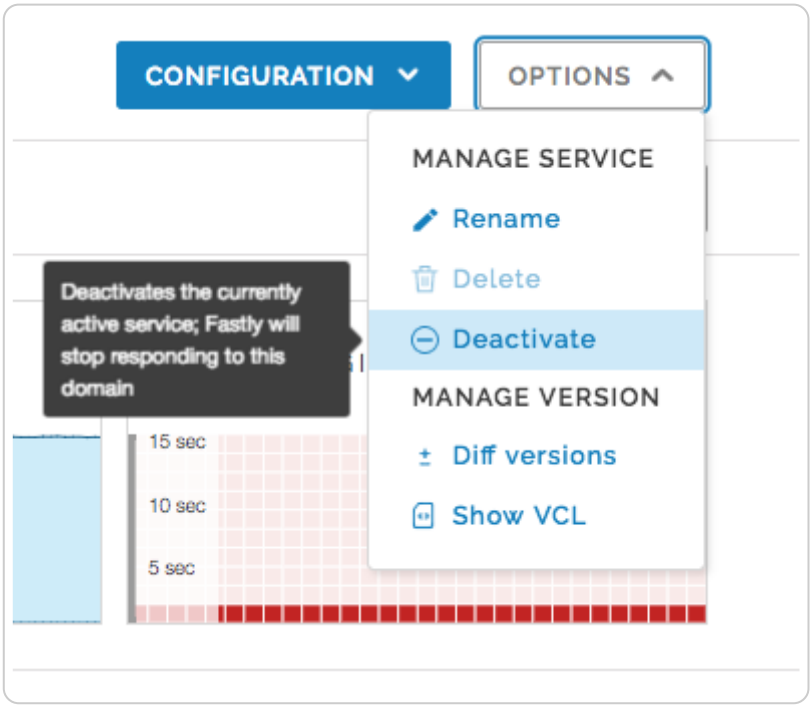


You can change the compared service versions by selecting a different version number in the selection menus.

Deactivating a service

To deactivate a service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Options** button to open the **Manage service** menu and select **Deactivate**.



❗ **IMPORTANT:** To minimize the risk of unauthorized use of your domains, we strongly recommend modifying or deleting any DNS CNAME records (/guides/basic-setup/adding-cname-records) pointing to the Fastly hostname associated with the deactivated service. Follow the instructions on your DNS provider's website.

You can also activate or deactivate a service via the API (/api/config#version). Did you accidentally delete a service? We can help.

Reactivating a service

To reactivate a service, follow the steps below:

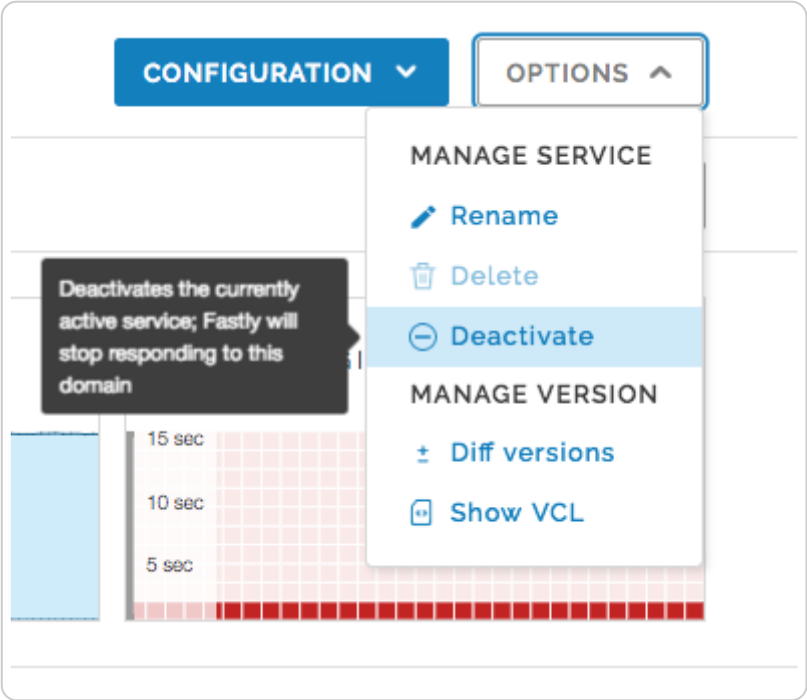
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click **Activate**. The service is reactivated.
5. If you removed the DNS CNAME records for the service's domains when you deactivated the service, you should add new DNS CNAME records (/guides/basic-setup/adding-cname-records) now.

Deleting a service

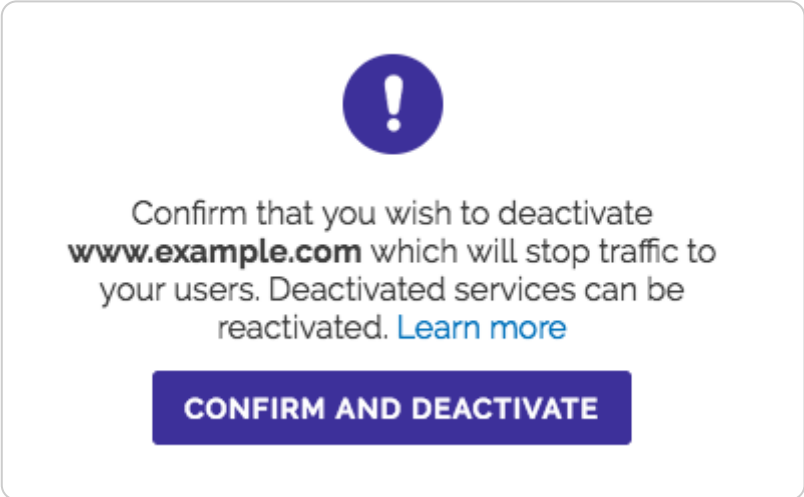
Fastly allows you to delete any service you create, along with all of its versions. Fastly does not offer a way to delete specific versions of a service, however. Service versions are meant to be an historic log of the changes that were made to a service. To undo changes introduced by a particular service version, you can always go back to a previous version and reactivate it or clone a new service version based on any old version.

To delete any service along with all of its versions, follow the steps below:

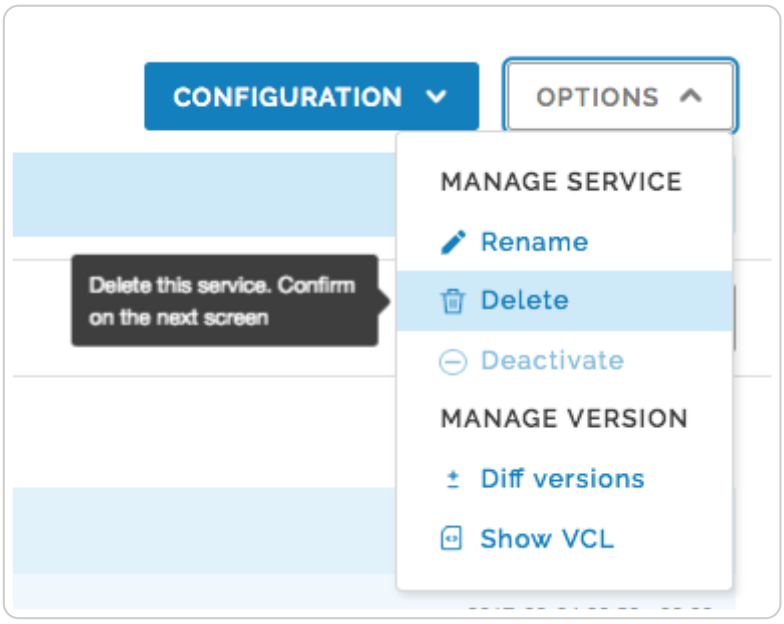
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Options** button to open the **Manage service** menu and select **Deactivate**.



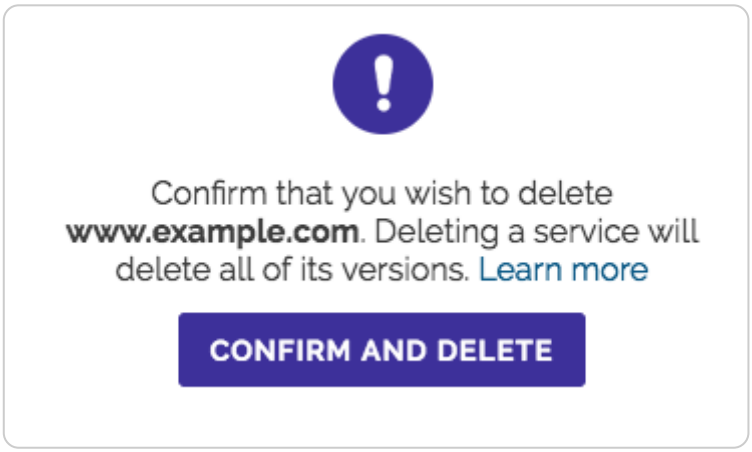
The deactivate service warning appears.



4. Click the **Confirm and deactivate** button to confirm you want to deactivate your service and acknowledge that you no longer want to serve traffic with it.
5. Click the **Options** button again to open the **Manage service** menu and select **Delete**.



The confirm delete window appears.



6. Click the **Confirm and delete** button to confirm that you want to delete the service.

IMPORTANT: To minimize the risk of unauthorized use of your domains, we strongly recommend modifying or deleting any DNS CNAME records (/guides/basic-setup/adding-cname-records) pointing to the Fastly hostname associated with the deleted service. Follow the instructions on your DNS provider's website.

Getting help with accidental service deletions

Services can be deactivated or deleted. Deactivated services can be reactivated at any time, but once they've been deleted you must contact Customer Support (mailto:support@fastly.com) to have them restored. When sending your request, remember to include:

- your customer ID (/guides/account-management-and-security/finding-and-managing-your-account-info#finding-your-customer-id)

- your company name
- your service ID (/guides/account-management-and-security/finding-and-managing-your-account-info#finding-your-service-id) (the name of the service you want restored)

Customer Support will notify you when your service has been restored.

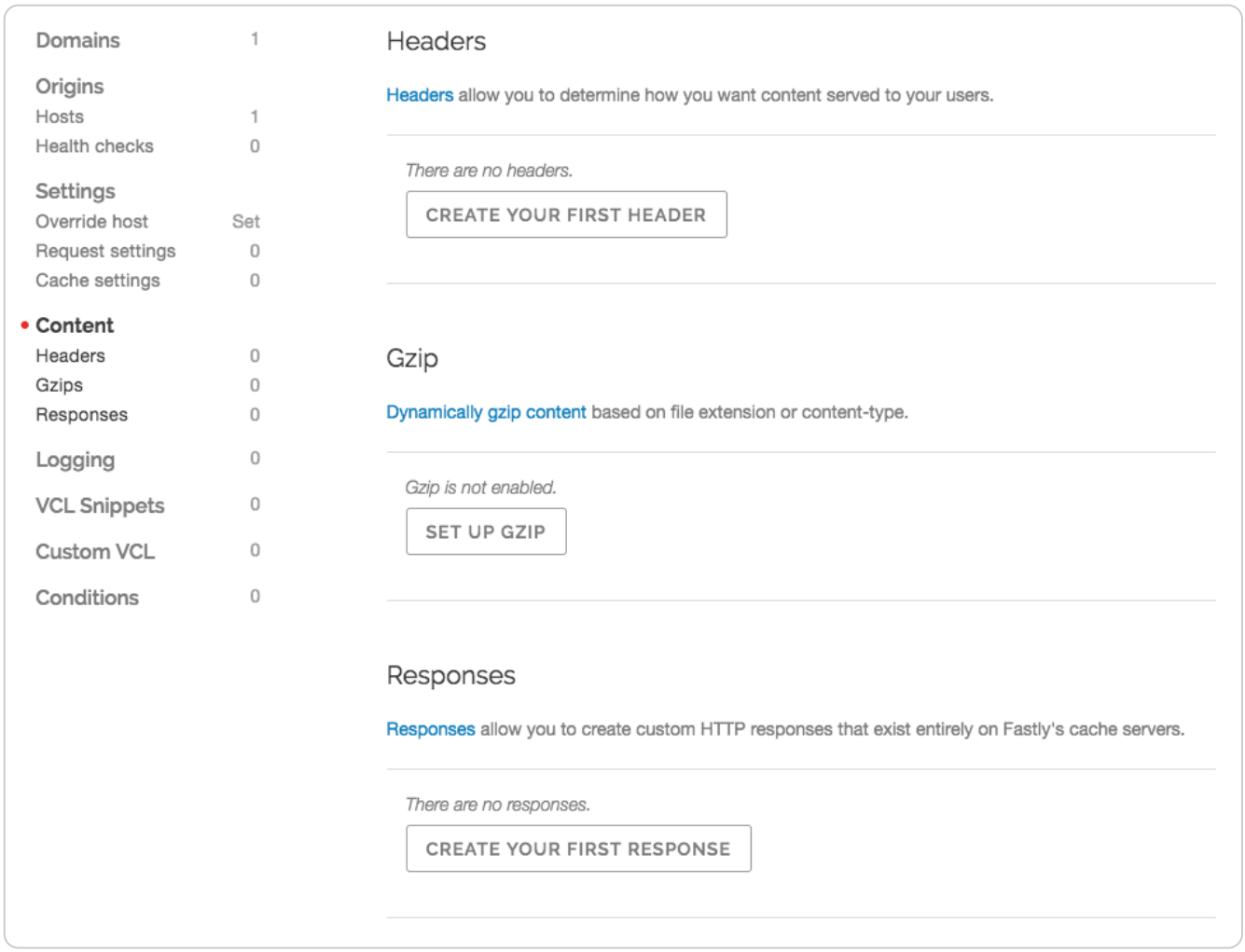
---

§ Adding or modifying headers on HTTP requests and responses (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses)

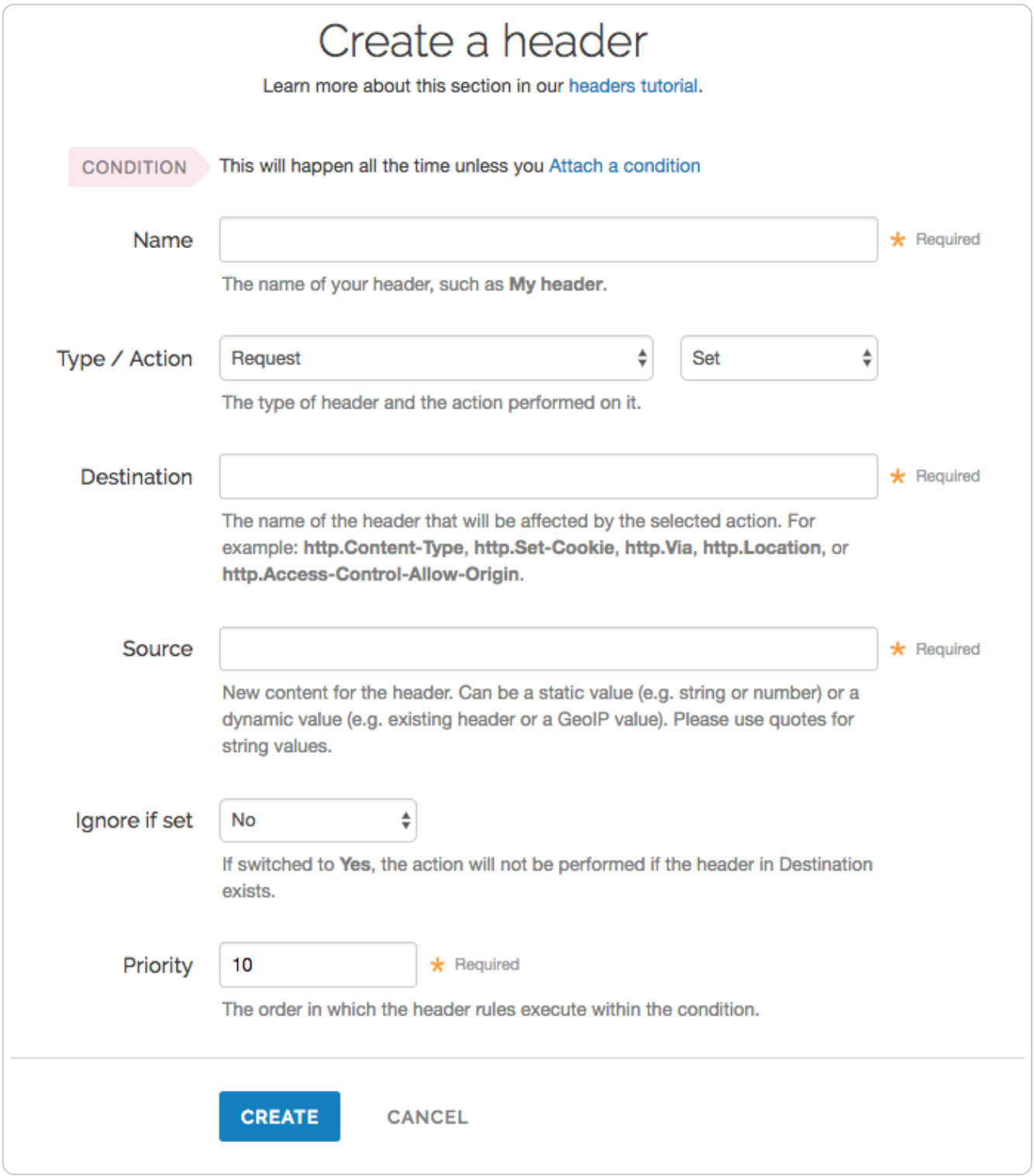
HTTP header fields are components of the header section of request and response messages in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction. When you create and configure headers, you can determine how you want your content served to your users. The following steps show you how to add and edit headers.

Create new headers

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `My header`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header affected by the selected action.
   - In the **Source** field, type where the content for the header comes from.

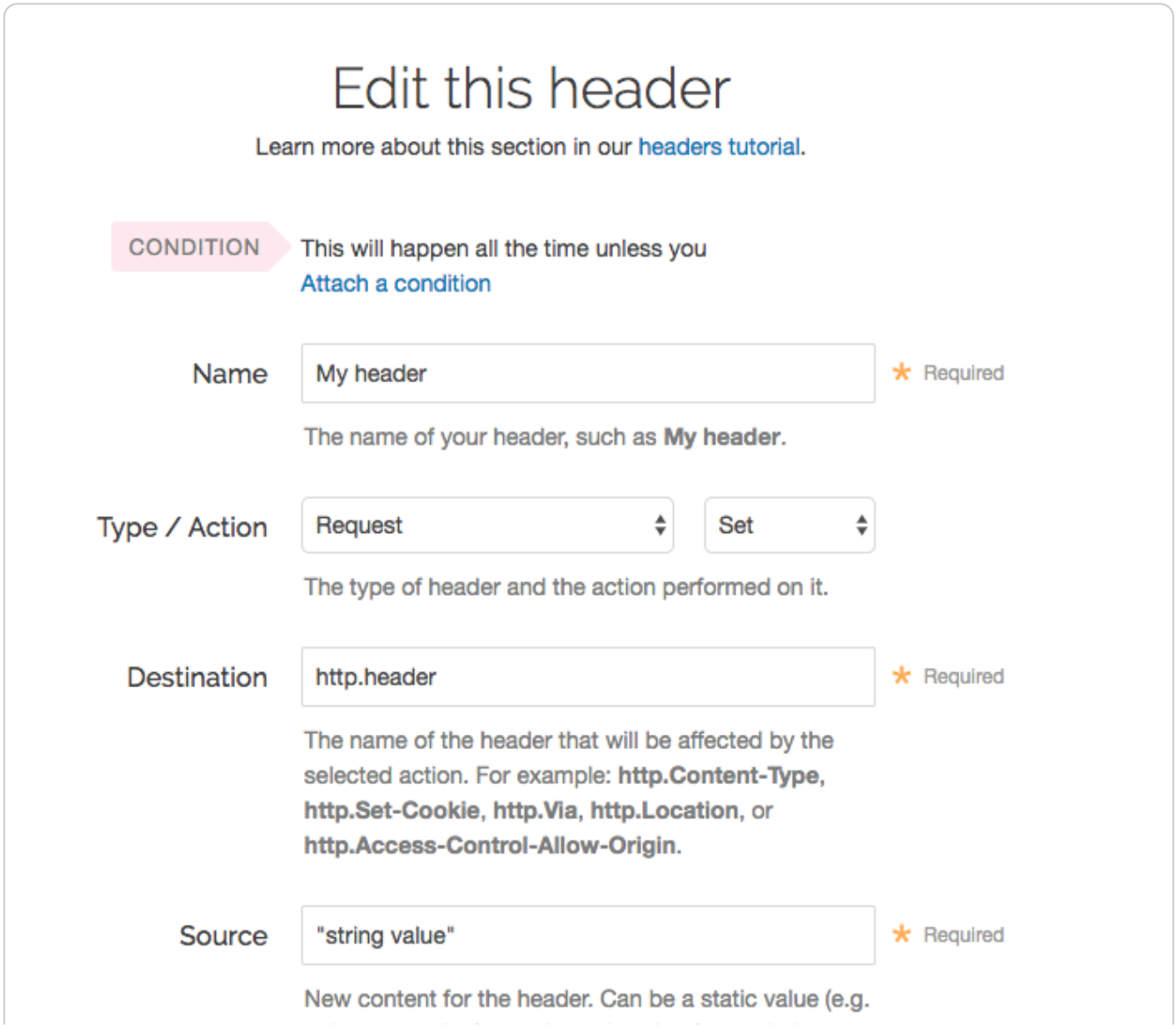
- From the **Ignore if set** menu, select **No** if you want the header in the **Destination** field modified or select **Yes** if you don't want it modified.
- In the **Priority** field, type the order the header rules execute.

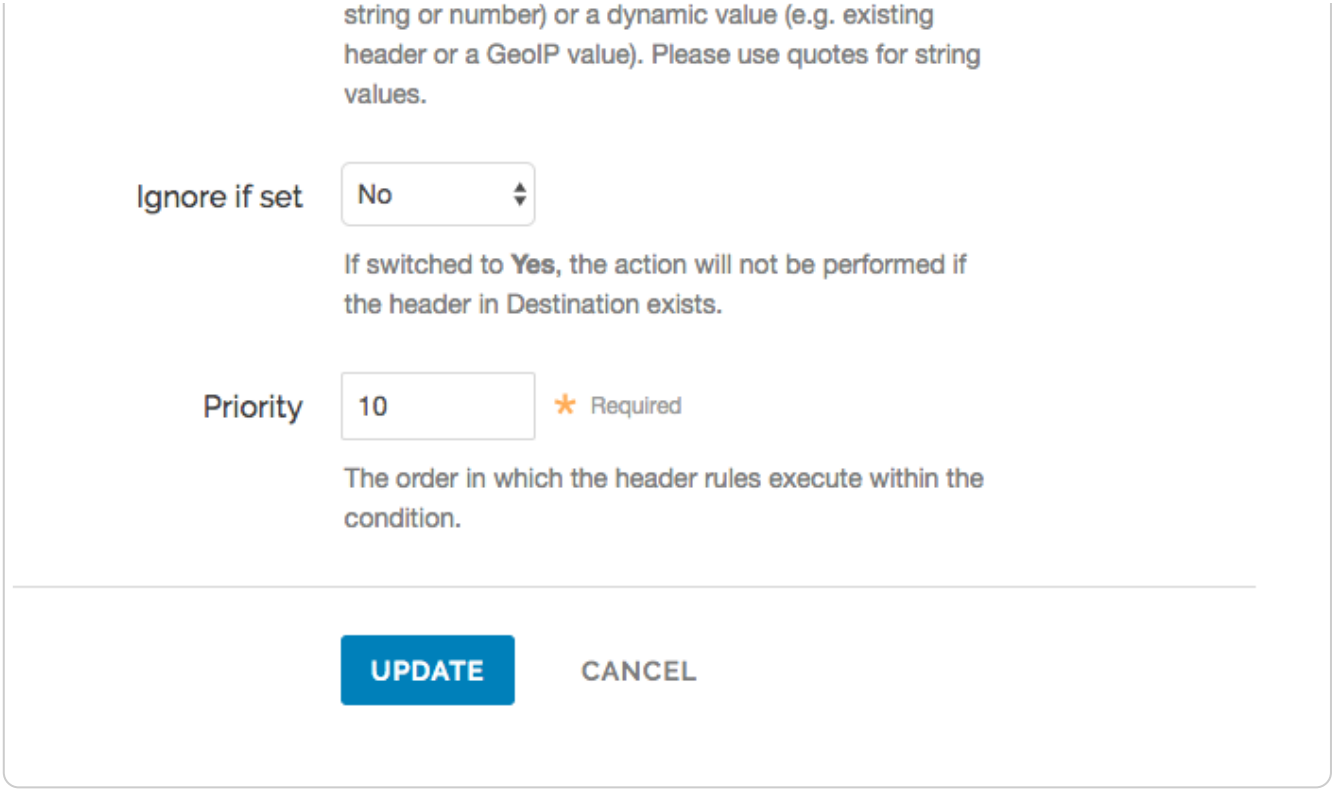
The Field description table below provides additional details about each of these controls.

7. Click the **Create** button.
8. Click the **Activate** button to deploy your configuration changes.

Edit headers

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the name of the header you want to edit. The Edit this header page appears.





6. Fill out the **Edit this header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `My header`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header affected by the selected action.
   - In the **Source** field, type where the content for the header comes from.
   - From the **Ignore if set** menu, select **No** if you want the header in the **Destination** field modified or select **Yes** if you don't want it modified.
   - In the **Priority** field, type the order the header rules execute.
7. Click the **Update** button.
8. Click the **Activate** button to deploy your configuration changes.

Field description table

This table describes what each field in the Header window means:

Field	Description
Name	The Name field specifies a memorable word or phrase that allows you to recognize and remember a particular Header rule.

Field	Description
Type	The Type menu can be set to **Request**, **Response**, or **Cache**. Selecting **Request** modifies the request coming from the user, and this will carry through to the request that gets sent to your origin server. Selecting **Response** affects the HTTP response that is sent back to the user. Selecting **Cache** affects the HTTP response that your origin server returns before it gets stored on Fastly servers, meaning whatever changes you make there will be remembered on a cache hit.
Action	The Action menu can be set to **Set**, **Append**, **Delete**, **Regex**, and **Regex All**. Selecting **Set** (the default) will write a value into the header (potentially overwriting it, if it already exists). Selecting **Append** will add a value onto the end of a header or set it if it doesn't exist. Selecting **Delete** will remove a header. When selected, it hides the Source field in the Header window. Selecting **Regex** allows you to perform a find and replace on specific text and is based on a regular expression you type in. When selected, the Regex and Substitution controls appear in the Header window. Selecting **Regex All** allows you to perform the same function as **Regex** but it performs a find and replace multiple times. When selected, the Regex and Substitution controls appear in the Header window.
Destination	The Destination field determines the name of the header that is going to be affected by our Action. Because header rules can be used to affect more than just HTTP headers, your input to this field should be formatted like this: `http.Header-Name`.
Source	The Source field is available on Set, Append, Regex, and Regex All actions. This field becomes hidden in the Header window when you select Delete from the Action menu. It determines where the new content for the header comes from. There are a plethora of options for Source. The simplest is a static string such as `"My Static String"` (including the quotes). Other options include `client.ip`, `req.http.Another-Header`, and `client.geo.city`. See the list of Common Sources below for more common sources of new content.
Regex	The Regex field only appears in the Header window when you select Regex or Regex All from the Action menu. It allows you to perform a find and replace on specific text and is based on a regular expression that you type in.
Substitution	The Substitution field only appears in the Header window when you select the Regex and Regex All from the Action menu. It replaces the text that was removed by the regex expression with the text you typed in the Substitution field.
Ignore if set	By default this is set to No, which means that if the header you are modifying already exists, it will be modified.

Field	Description
Priority	The Priority field determines the order in which the header rules execute (e.g., a priority of 1 means the header rule executes first). This can be important if you set headers and then set other headers based on the earlier ones.

Common sources of new content

Name	Valid Types	Description
`req.http.Fastly-Client-IP`	Request, Cache, Response	The true IP address of the client.
`client.ip` and `client.identity`	Request, Cache, Response	The client IP address. These variables are available, but may not always display the source IP address. For instance, they may show the edge node IP when shielding is enabled. For the true client IP address use `req.http.Fastly-Client-IP`. **IMPORTANT:** In some cases, client IP data may be considered sensitive. Make sure you protect the sensitive IP data you stream or store.
`server.identity`	Request, Cache, Response	A unique identifier for the Fastly server processing the request.
`server.region`	Request, Cache, Response	The region in which the Fastly server resides.
`server.datacenter`	Request, Cache, Response	The datacenter in which the Fastly server resides.
`req.url`	Request, Cache, Response	The URL of the HTTP Request from the client.
`req.http.*`	Request, Cache, Response	The headers from the HTTP Request, access as: `req.http.HeaderName`
`beresp.status`	Cache	The status returned from the origin server.

Name	Valid Types	Description
`beresp.http.*`	Cache	The headers from the origin's HTTP Response, access: `beresp.http.HeaderName`
`resp.status`	Response	The status that is going to be returned to the client.
`resp.http.*`	Response	The headers in the HTTP Response to be returned to the client, access: `resp.http.HeaderName`
`client.geo.*`	Request, Cache, Response	Geolocation values for the client's IP (see our geolocation article (/guides/vcl/geolocation-related-vcl-features) for more information).

§ Caching configuration best practices (/guides/basic-configuration/caching-best-practices)

To ensure optimum origin performance during times of increased demand or during scheduled downtime for your servers, consider the following best practices for your service's caching configurations.

Check your cache hit ratio

The number of requests delivered by a cache server, divided by the total number of requests, is called the "cache hit ratio." A high cache hit ratio means you've kept request traffic from hitting your origin unnecessarily; requests come from cache instead. In general, you want your cache hit ratio as high as possible, usually in excess of 90%. You can check your hit ratio by viewing the Stats page (/guides/basic-concepts/about-the-web-interface-controls#about-the-stats-page) for your service.

Set a Fallback TTL

The amount of time information can be retained in cache memory is considered its "time to live" (/guides/detailed-product-descriptions/about-fastlys-full-site-delivery-features#time-to-live-support) or TTL. Your origin server sets the TTL for a variety of objects automatically. When no TTL exists for an object, however, you can specifically set a fallback TTL (sometimes called a "default TTL").

By default, we set your fallback TTL to 3600 seconds (60 minutes). This gives you time to diagnose the root cause of unexpected errors and then purge information from cache once those issues are resolved. You can update that TTL at any time as follows:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. In the **Fallback TTL** area, click the pencil icon next to the TTL setting.
6. In the **Change the Fallback TTL** field, type the new TTL in seconds.
7. Click **Save** to save your changes.

ℹ **NOTE:** See our Google Cloud Storage instructions if you're changing the default TTL for a GCS bucket (/guides/integrations/google-cloud-storage#changing-the-default-ttl-for-your-gcs-bucket).

Configure Fastly to temporarily serve stale content

If your origin becomes unavailable for an extended period of time (for example, being taken offline for maintenance purposes), temporarily serving stale content may help you. Serving stale content can also benefit you if your site's static content is updated or published quite frequently.

You can instruct Fastly to serve stale content by adding a `stale-while-revalidate` or `stale-if-error` statement on your `Cache-Control` or `Surrogate-Control` headers. Our guide to serving stale content (/guides/performance-tuning/serving-stale-content) describes this in more detail.

Decrease your first byte timeout time

After you have configured Fastly to temporarily serve stale, decreasing your first byte timeout time will cause stale content to be served to the requestor faster while fetching fresh content from the origin. Decreasing your first byte timeout time as well as serving stale will reduce unnecessary 503 first byte timeout errors. Decrease the first byte timeout time to your origin as follows:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. In the **Hosts** area, find your origin server and click the pencil icon to edit the host. The Edit this host page appears.

6. Click the **Advanced options** link at the bottom of the page. The Advanced options controls appear.
7. In the **First byte timeout** field, type the new first byte timeout in milliseconds. Approximately 15000 milliseconds is a good default to start with.
8. Click **Update** to save your changes.

Increase cache control header times

During times of increased demand, you can instruct Fastly to keep objects in cache as long as possible by increasing the times you set on your cache control headers. Consider changing the `max-age` on your `Cache-Control` or `Surrogate-Control` headers. Our guide to changing caching times on backend headers (/guides/tutorials/cache-control-tutorial) describes this in more detail.

Consider custom error handling

When downtime can't be avoided, standard error messages might not ensure the best user experience. Consider creating custom error messages that include information specific to the request being made and pertinent to the user. Our guide to creating error pages (/guides/basic-configuration/creating-error-pages-with-custom-responses) with custom responses provides more detail.

Inform Fastly Customer Support

We like to be sure we're readily available for assistance during customer events. When you know in advance that an event is forthcoming, contact support (/guides/detailed-product-descriptions/about-fastlys-customer-support#tips-on-what-to-include-in-a-support-request) with details. Be sure to include details about:

- the date and time of the event
- the type of event happening
- how long you expect it to last (if it's planned)
- the Fastly services that might be affected

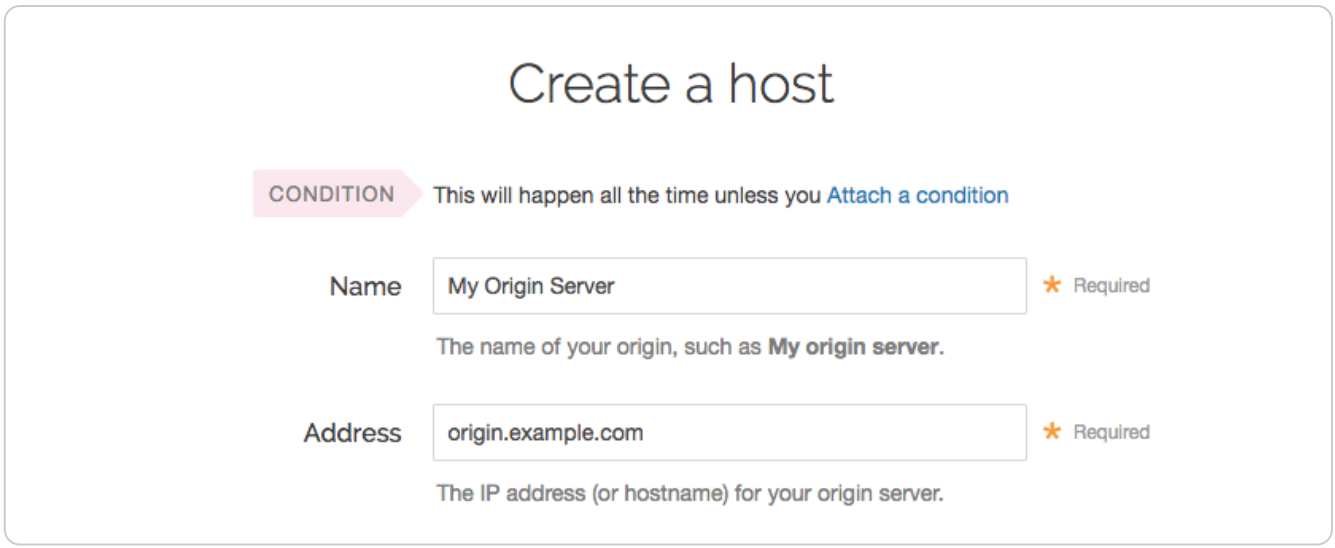
If the event you're planning is designed to validate the security of your service behind Fastly, be sure to read our guide to penetration testing (/guides/monitoring-and-testing/penetration-testing-your-service-behind-fastly) first.

---

§ Connecting to origins (/guides/basic-configuration/connecting-to-origins)

To communicate with your origin servers, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **Create host** button. The Create a host page appears.



6. Fill out the **Create a host** fields as follows:
   - In the **Name** field, type the name of your server (for example, `My Origin Server`). This name is displayed in the Fastly web interface.
   - In the **Address** field, type the IP address (or hostname) for your website's origin server.

   See Understanding the difference between certificate hostname and SNI hostname values for more information about hostnames.

RECOMMENDED Transport Layer Security (TLS)

Enabling TLS ensures a private and unmodified connection between Fastly and your origin server. Fastly automatically enables TLS for hosts set to port 443. Learn more about connecting to origins over TLS.

Enable TLS?

- (•) **Yes,** enable TLS and connect securely using port 443
- ( ) **No, do not enable TLS.** Instead connect using port 80

Verify certificate?

- (•) **Yes,** verify the authenticity of the TLS certificate
- ( ) **No,** do not verify my TLS certificate

Certificate hostname

* Required

This value is matched against the certificate common name (CN) or a subject alternate name (SAN).

› Need help looking up your Certificate hostname?

› What happened to SSL hostname?

SNI hostname

[ ] Match the SNI hostname to the Certificate hostname. This value identifies which certificate is to be used for the request to origin.

› When is SNI hostname required?

TLS CA certificate

CA certificate used to verify the certificate from origin. Must be in PEM form.

› When should I specify TLS CA certificate?

› Advanced TLS options — Minimum TLS version, Maximum TLS version, Ciphersuites, TLS client certificate, TLS client key

7. Fill out the **Transport Layer Security (TLS)** area as follows:
    - Leave the **Enable TLS?** default set to **Yes** if you want to enable TLS to secure the connection between Fastly and your origin. To enable TLS, a valid SSL certificate must be installed on your origin server and port 443 (or the specified port) must be open in the firewall. You can select **No** if you do not want to use TLS.
    - Leave the **Verify certificate?** default set to **Yes** if you want to verify the authenticity of the TLS certificate. Selecting **No** means the certificate will not be verified.

      > ⚠ **WARNING:** Not verifying the certificate has serious security implications, including vulnerability to man-in-the-middle attack. Consider uploading a CA certificate instead of disabling certificate validation.

    - In the **Certificate hostname** field, type the hostname associated with your TLS certificate. This value is matched against the certificate common name (CN) or a subject alternate name (SAN) depending on the certificate you were issued.
    - If you are specifying an SNI hostname, see the section below.
    - If you are specifying a TLS CA certificate, see the section below.
8. Fill out the remaining **Create a host** fields as follows:
    - From the **Shielding** menu, optionally select a POP to enable the shielding feature. For more information, see our guide on shielding (/guides/performance-tuning/shielding).
    - From the **Health check** menu, optionally select a health check for this origin server. For more information, see our health checks tutorial (/guides/basic-configuration/health-checks-tutorial).
    - From the **Auto load balance** menu, optionally select **Yes** to enable load balancing for this origin server. For more information, see our guide on load balancing (/guides/performance-tuning/load-balancing-configuration).
    - If you enabled load balancing, type a weight in the **Weight** field.
9. Click the **Create** button.
10. Click the **Activate** button to deploy your configuration changes.

And that's all you need to do. Everything else is optional, but just in case you'd like to set them, we've included the information below.

Setting the TLS hostname

Normally we check the server certificate against the hostname portion of the address for your origin entered in the **Create a host** window. Checking the certificate is done by using the value of the Certificate Hostname field in your origin TLS settings. To have Fastly verify the certificate using

a different hostname, specify it via the **SNI Hostname** field under **Advanced options**.

This information also gets sent to the server in the TLS handshake. If you are using Server Name Indication (https://en.wikipedia.org/wiki/Server_Name_Indication) (SNI) to put multiple certificates on your origin, specifying it in the **SNI Hostname** field will select which one is used.

Understanding the difference between certificate hostname and SNI hostname values

The following explains the difference between a certificate and SNI hostname value:

The certificate hostname (ssl_cert_hostname). This hostname validates the certificate at origin. This value should match the certificate common name (CN) or an available subject alternate name (SAN). It displays as ssl_cert_hostname in VCL. This doesn't affect the SNI certification. You can set this value in **Certificate hostname** field of the **TLS options** page.

The SNI hostname (ssl_sni_hostname). This hostname determines which certificate should be used for the TLS handshake. SNI is generally only required when your origin is using shared hosting, such as Amazon S3, or when you use multiple certificates at your origin. SNI allows the origin server to know which certificate to use for the connection. This value displays as ssl_sni_hostname in VCL. This doesn't affect the certificate validation.

If you don't enter an actual value in your certificate hostname, the .host value is used by default to verify the certificate. The .host value is the actual IP address or virtual hostname you enter in the **Address** field on the **Host** area of the **Origins** page. This value is matched against the certificate common name (CN) or a subject alternate name (SAN).

The table below shows you what happens when you set the Certificate and SNI hostname values in the TLS settings:

If Certificate hostname contains…	**and SNI hostname contains…**	**then the Certificate Validation value will be…**	**and the SNI value will be…**
www.example.com	nothing	www.example.com	nothing
nothing	www.example.org	the .host value from the **Address** field	www.example.org
www.example.com	www.example.org	www.example.com	www.example.org
nothing	nothing	the .host value from the **Address** field	nothing

About the ssl_hostname value (deprecated). The ssl_hostname value has been deprecated and replaced with ssl_cert_hostname and ssl_sni_hostname. Use these two values instead.

IMPORTANT: If you use an IP address for your .host value (i.e., by not entering a value in your certificate hostname), this will generate an error (/guides/debugging/common-503-errors#error-503-hostname-doesnt-match-against-certificate) where the certificate hostname specified in your service's origin TLS settings doesn't match either the Common Name (CN) or available Subject Alternate Names (SANs).

Specifying a TLS CA certificate

If you're using a certificate that is either self-signed or signed by a Certificate Authority (CA) not commonly recognized by major browsers (and unlikely to be in the Ubuntu bundle that we use), you can provide the certificate in PEM format via the **TLS CA certificate** field. The PEM format looks like this:

```
-----BEGIN CERTIFICATE-----
MIIDrzCCApegAwIBAgIQCDvgVpBCRrGhdWrJWZHHSjANBgkqhkiG9w0BAQUFADBh
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSAwHgYDVQQDExdEaWdpQ2VydCBHbG9iYWwgUm9vdCBD
QTAeFw0wNjExMTAwMDAwMDBaFw0zMTExMTAwMDAwMDBaMGExCzAJBgNVBAYTAlVT
MRUwEwYDVQQKEwxEaWdpQ2VydCBJbmMxGTAXBgNVBAsTEHd3dy5kaWdpY2VydC5j
b20xIDAeBgNVBAMTF0RpZ2lDZXJ0IEdsb2JhbCBSb290IENBMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4jvhEXLeqKTTo1eqUKKPC3eQyaKl7hLOllsB
CSDMAZOnTjC3U/dDxGkAV53ijSLdhwZAAIEJzs4bg7/fzTtxRuLWZscFs3YnFo97
nh6Vfe63SKMI2tavegw5BmV/Sl0fvBf4q77uKNd0f3p4mVmFaG5cIzJLv07A6Fpt
43C/dxC//AH2hdmoRBBYMql1GNXRor5H4idq9Joz+EkIYIvUX7Q6hL+hqkpMfT7P
T19sdl6gSzeRntwi5m3OFBqOasv+zbMUZBfHWymeMr/y7vrTC0LUq7dBMtoM1O/4
gdW7jVg/tRvoSSiicNoxBN33shbyTApOB6jtSj1etX+jkMOvJwIDAQABo2MwYTAO
BgNVHQ8BAf8EBAMCAYYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUA95QNVbR
TLtm8KPiGxvDl7I90VUwHwYDVR0jBBgwFoAUA95QNVbRTLtm8KPiGxvDl7I90VUw
DQYJKoZIhvcNAQEFBQADggEBAMucN6pIExIK+t1EnE9SsPTfrgT1eXkIoyQY/Esr
hMAtudXH/vTBH1jLuG2cenTnmCmrEbXjcKChzUyImZOMkXDiqw8cvpOp/2PV5Adg
06O/nVsJ8dWO41P0jmP6P6fbtGbfYmbW0W5BjfIttep3Sp+dWOIrWcBAI+0tKIJF
PnlUkiaY4IBIqDfv8NZ5YBberOgOzW6sRBc4L0na4UU+Krk2U886UAb3LujEV0ls
YSEY1QSteDwsOoBrp+uvFRTp2InBuThs4pFsiv9kuXclVzDAGySj4dzp30d8tbQk
CAUw7C29C79Fv1C5qfPrmAESrciIxpg0X40KPMbp1ZWVbd4=
-----END CERTIFICATE-----
```

Specifying a TLS client certificate and key

To ensure TLS connections to your origin come from Fastly and aren't random, anonymous requests, set your origin to verify the client using a client certificate. Simply paste the certificate and private key in PEM form into the appropriate text boxes on the **TLS options** page.

IMPORTANT: The private key must not be encrypted with a passphrase.

Then configure your backend to require client certificates and verify them against the CA cert they were signed with. Here are some ways of doing that:

- Apache (http://httpd.apache.org/docs/2.4/ssl/ssl_howto.html#accesscontrol)
- Nginx (http://nginx.org/en/docs/http/ngx_http_ssl_module.html#ssl_client_certificate)

- IIS (https://ondrej.wordpress.com/2010/01/24/iis-7-and-client-certificates/)

Specifying acceptable TLS protocol versions

If your origin server is configured with support for modern TLS protocol versions, you can customize the TLS protocols Fastly will use to connect to it by setting a **Minimum TLS Version** and **Maximum TLS Version** under **Advanced options**. We recommend setting both to the most up-to-date TLS protocol, currently 1.2, if your origin can support it.

Use the `openssl` command to verify your origin supports a given TLS protocol version. For example:

```
openssl s_client -connect origin.example.com:443 -tls1_2
```

Replace `-tls1_2` with `tls1_1` and `tls1_0` to test other protocol versions. Fastly does not support SSLv2 or SSLv3.

Specifying acceptable TLS cipher suites

Fastly supports configuring the OpenSSL cipher suites used when connecting to your origin server. This allows you to turn specific cipher suites on or off based on security properties and origin server support. The **Ciphersuites** setting under **Advanced options** accepts an OpenSSL formatted cipher list (https://www.openssl.org/docs/manmaster/man1/ciphers.html). We recommend using the strongest cipher suite your origin will support as detailed by the Mozilla SSL Configuration Generator (https://mozilla.github.io/server-side-tls/ssl-config-generator/).

Use the `openssl` command to verify your origin supports a given cipher suite. For example:

```
openssl s_client -connect origin.example.com:443 -tls1_2 -cipher ECDHE-RSA-AES128-GCM-SHA256
```

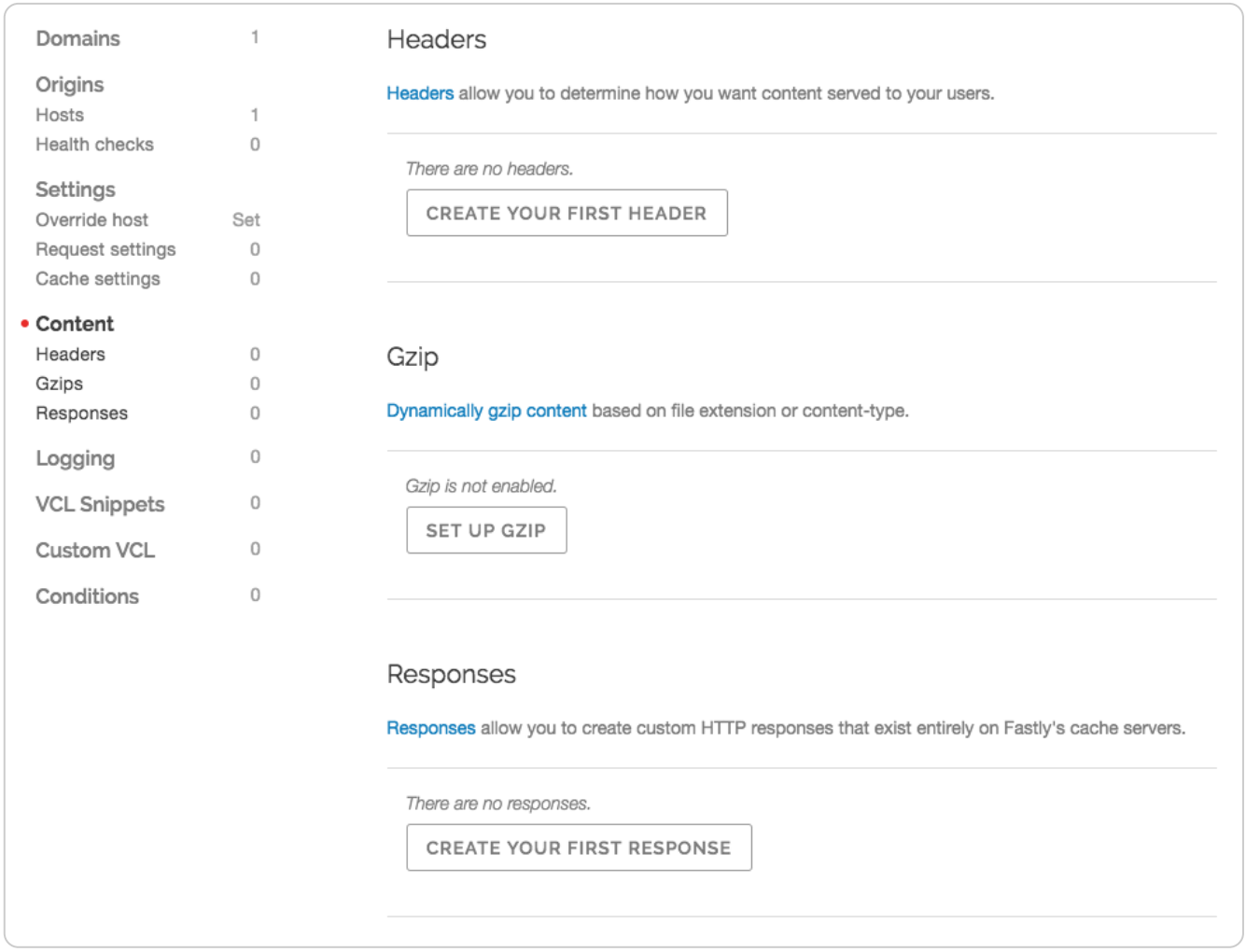
Replace `-cipher ECDHE-RSA-AES128-GCM-SHA256` with the cipher suite to test.

---

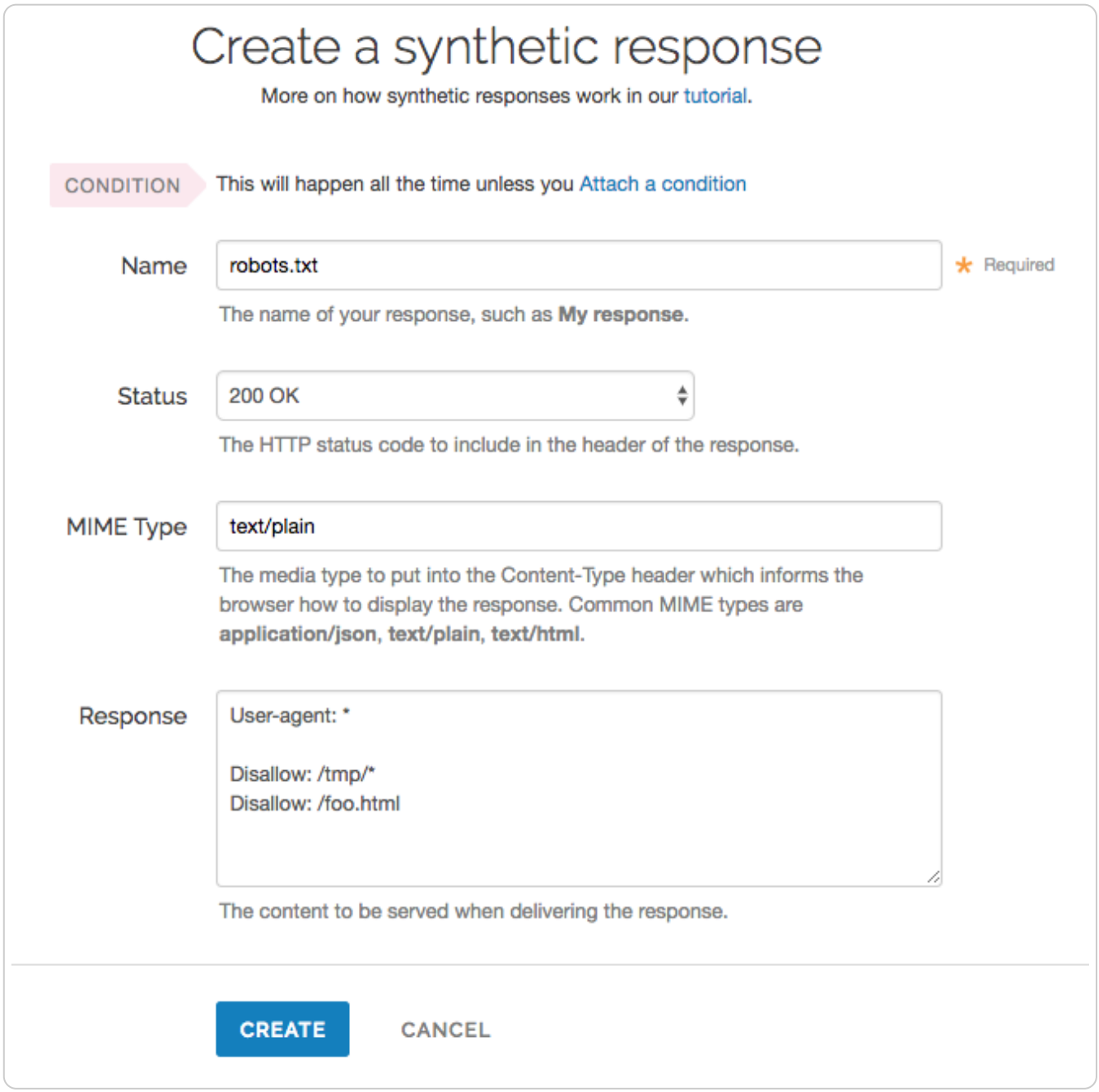
§ Creating and customizing a robots.txt file (/guides/basic-configuration/creating-and-customizing-a-robots-file)

Once you've created a CNAME DNS record (/guides/basic-setup/adding-cname-records) to direct your domain's traffic to Fastly, you can set custom responses (/guides/vcl/custom-responses-that-dont-hit-origin-servers) that will be served by the robots.txt file (https://en.wikipedia.org/wiki/Robots_exclusion_standard) on your website. To create and configure your robots.txt file via the Fastly web interface, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.

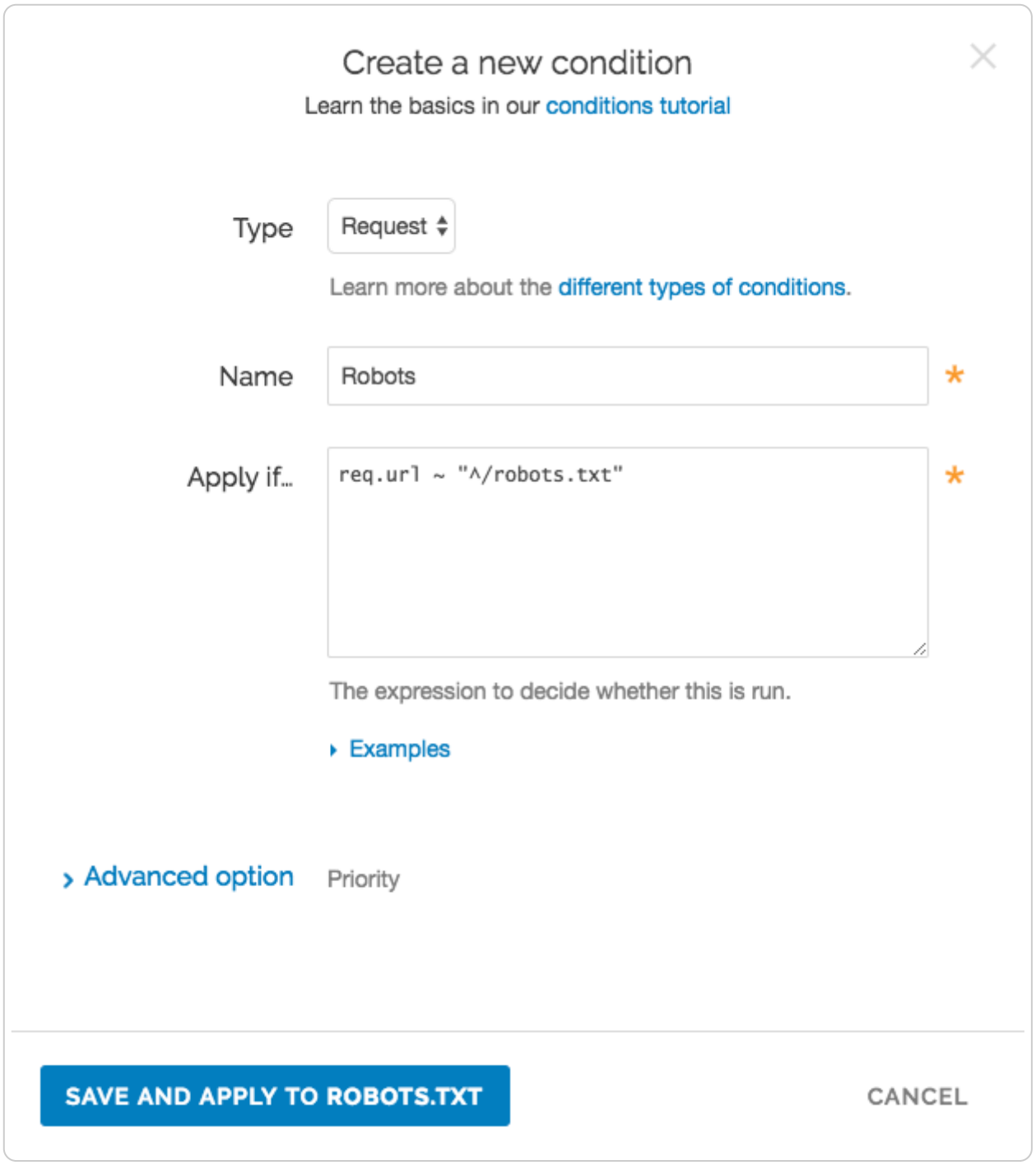


5. Click the **Create response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type an appropriate name. For example `robots.txt`.
   - Leave the **Status** menu set at its default `200 OK`.
   - In the **MIME Type** field, type `text/plain`.
   - In the **Response** field, type at least one User-agent string and one Disallow string. For instance, the above example tells all user agents (via the `User-agent: *` string) they are not allowed to crawl anything after `/tmp/` directory or the `/foo.html` file (via the `Disallow: /tmp/*` and `Disallow: /foo.html` strings respectively).
7. Click the **Create** button. Your new response appears in the list of responses.

8. Click the **Attach a condition** link to the right of the newly created response. The Create a new condition window appears.



9. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select the desired condition (for example, `Request`).
   - In the **Name** field, type a meaningful name for your condition (e.g., `Robots`).
   - In the **Apply if** field, type the logical expression to execute in VCL to determine if the condition resolves as true or false. In this case, the logical expression would be the location of your robots.txt file (e.g., `req.url ~ "^/robots.txt"`).

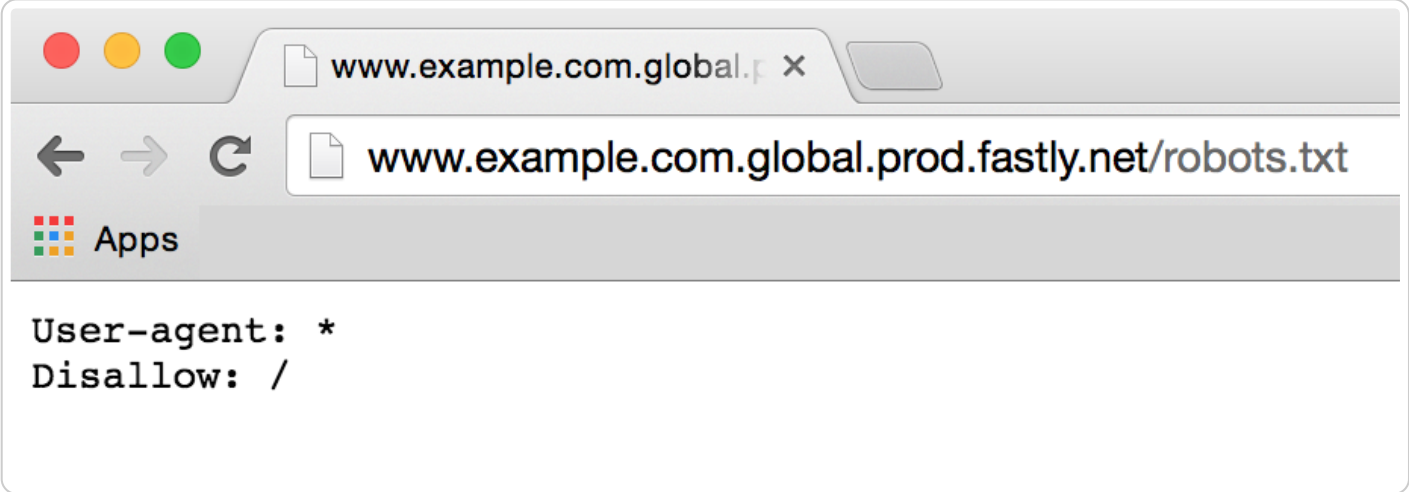
10. Click the **Save and apply to** button.
11. Click the **Activate** button to deploy your configuration changes.

> ℹ **NOTE:** For an in-depth explanation of creating custom responses, check out our Responses Tutorial (/guides/basic-configuration/responses-tutorial).

Why can't I customize my robots.txt file with global.prod.fastly.net?

Adding the `.global.prod.fastly.net` extension to your domain (for example, `www.example.com.global.prod.fastly.net`) via the browser or in a cURL command can be used to test how your production site will perform using Fastly's services.

To prevent Google from accidentally crawling this test URL, we provide an internal robots.txt file that instructs Google's webcrawlers to ignore all pages for all hostnames that end in `.prod.fastly.net`.



This internal robots.txt file cannot be customized via the Fastly web interface until after you have set the CNAME DNS record for your domain to point to `global.prod.fastly.net`.

---

§ Creating error pages with custom responses (/guides/basic-configuration/creating-error-pages-with-custom-responses)

The default error responses served by Fastly can be jarring for your users, especially when using Fastly for consumer applications. This tutorial shows you how to set up your service configuration to serve a custom page or a synthetic response when we receive an error code from your backend.

> ★ **TIP:** Fastly can optionally serve stale content when there is a problem with your origin server. For more information, see our guide on serving stale content (/guides/performance-tuning/serving-stale-content).

We assume you are already accustomed to editing and deploying configurations using the web interface (https://manage.fastly.com/). If you are not familiar with the web interface, see our basic setup (/guides/basic-setup/) and configuration (/guides/basic-configuration/) information to learn more before you continue.

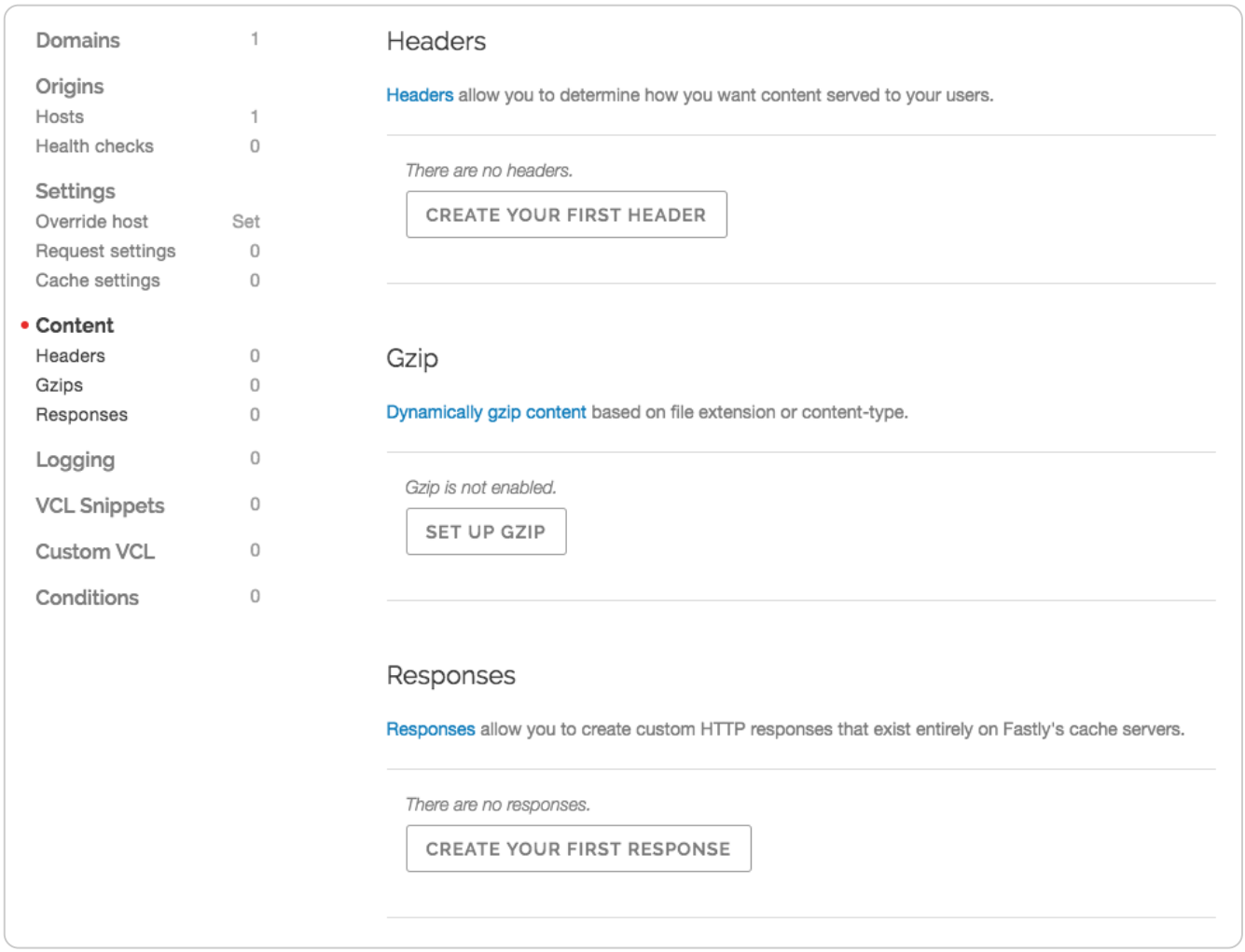
Overview

In this tutorial, we will create a response object that contains the HTML you want to serve for your error page. We provide example HTML, but you can use any HTML you see fit. The response object will require that you use a condition in order for it to be served. If you are not familiar with conditions or responses we suggest you read the following:

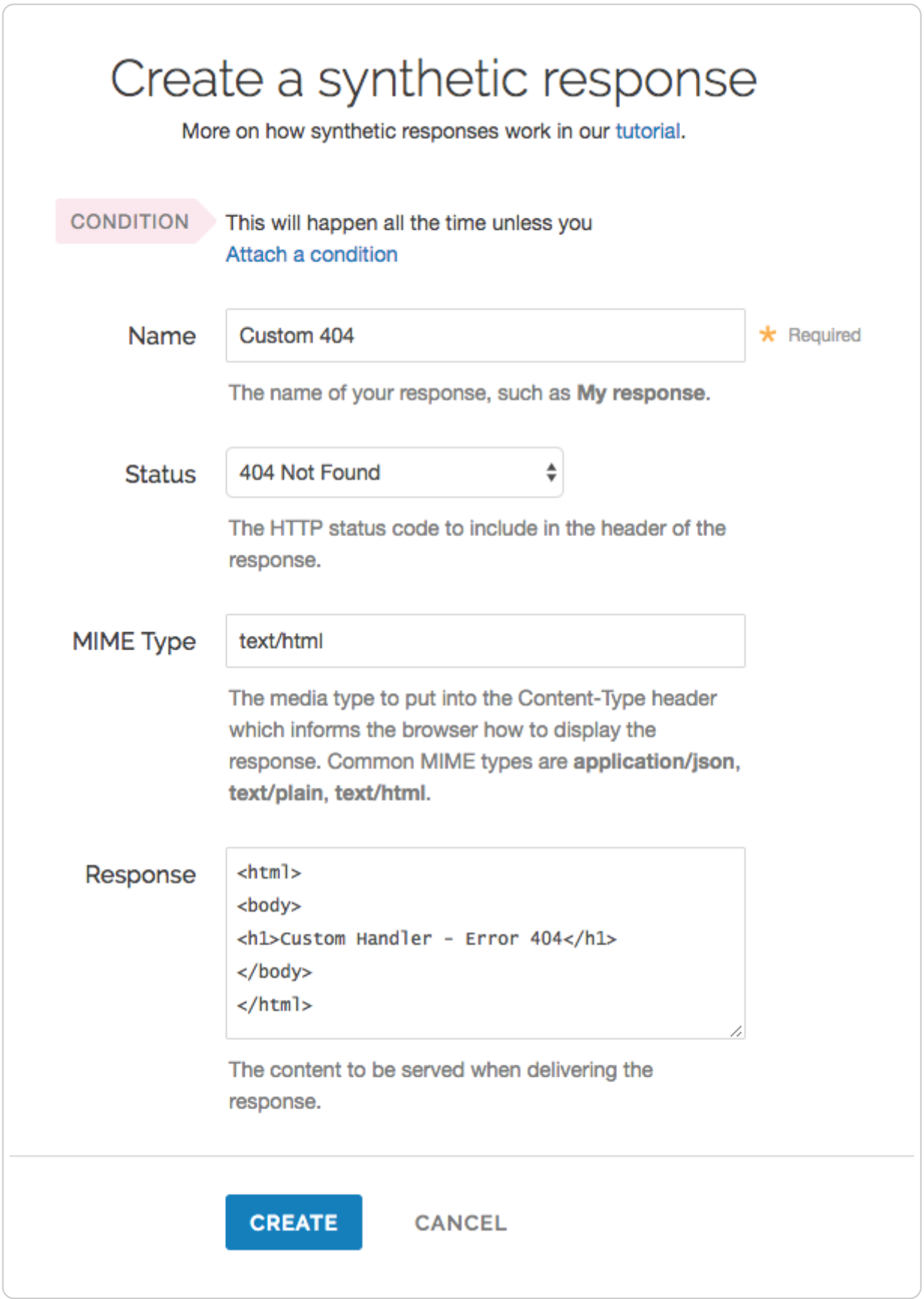
- Responses Tutorial (/guides/basic-configuration/responses-tutorial)
- Conditions guides (/guides/conditions/)

Creating the custom response

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



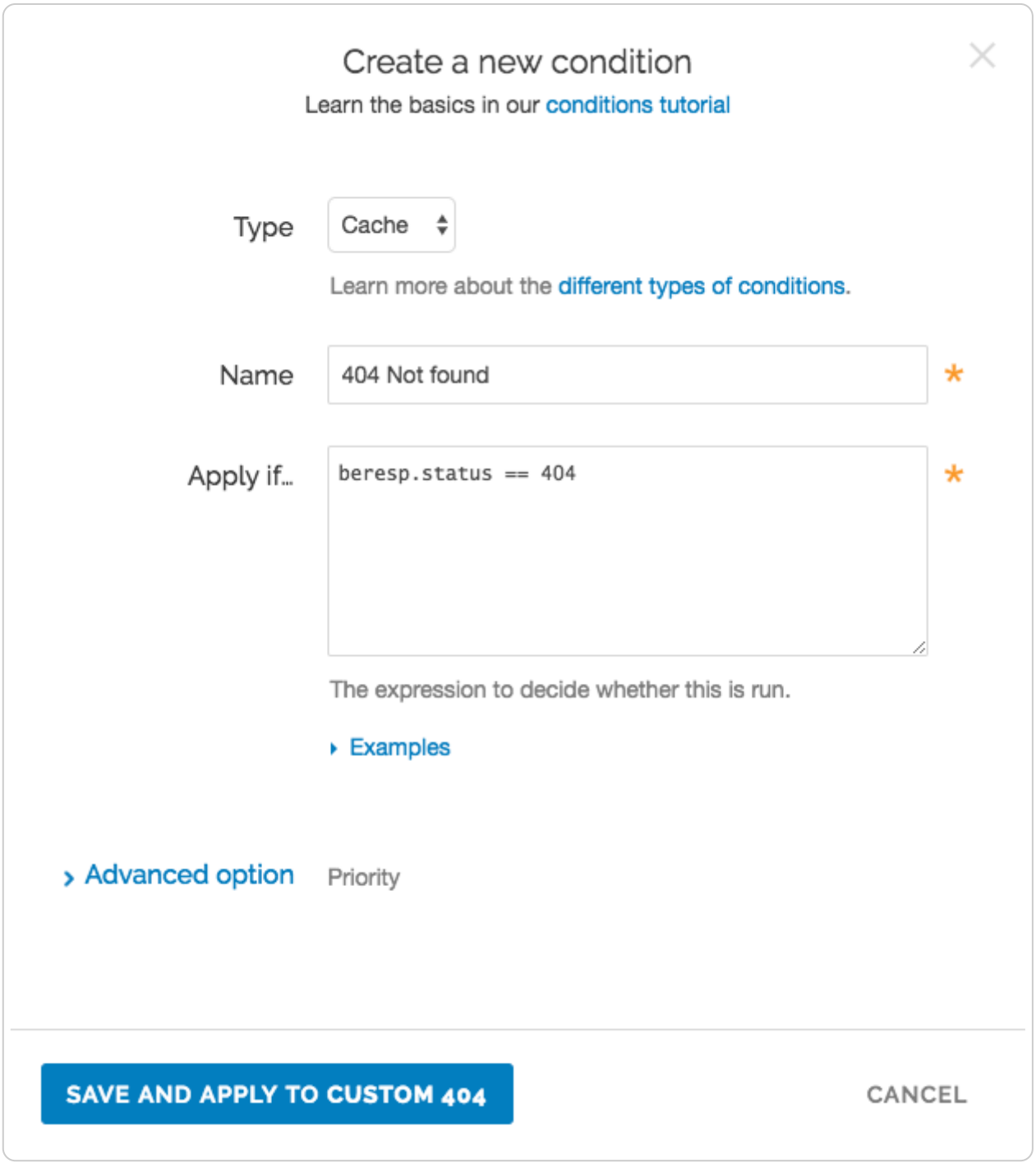
5. Click the **Create response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type a name for the response you're creating (e.g., `Custom 404`).
   - From the **Status** menu, select the appropriate status (e.g., `404 Not Found`).

- In the **MIME Type** field, specify the Content-Type of the response (e.g., `text/html`).
- In the **Response** field, type the content to be served when delivering a response.

7. Click the **Create** button. Your new response appears in the list of responses.
8. Click the **Attach a condition** link to the right of the name of your new response. The Create a new condition window appears.



9. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select the type of condition you want (e.g., `Cache`).
   - In the **Name** field, type a name for the condition you're creating (e.g., `404 Not Found`).

- In the **Apply if** field, type the condition under which the new response occurs in the following format:

  `beresp.status == ###`

  where ### equals the status condition you're creating the response for. For example, using the value of `beresp.status == 404` in the **Apply if** field here tells Fastly to use this response object whenever origin servers return a 404 status. (See the Conditions guides (/guides/conditions/) for more detailed information on conditions.)

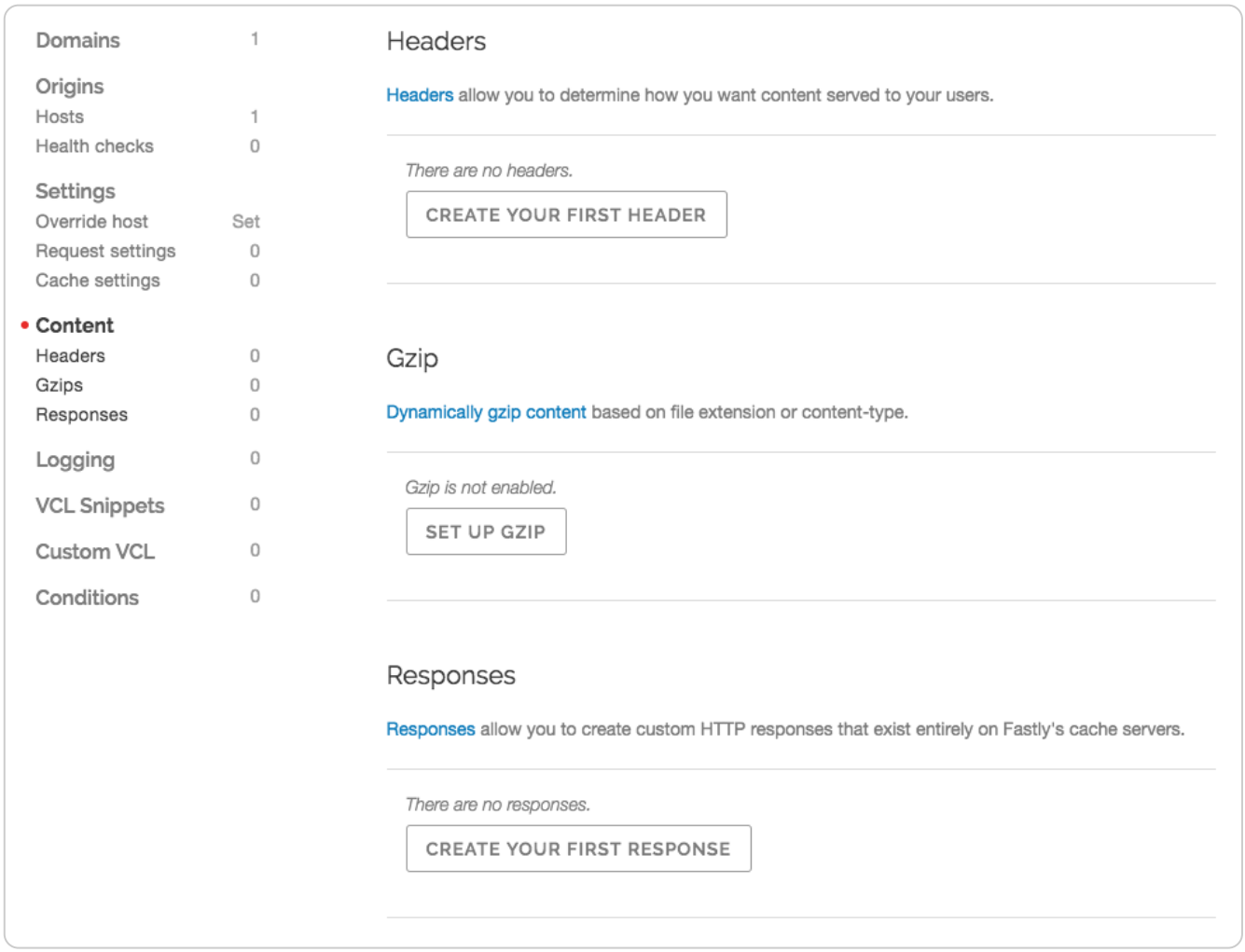
10. Click the **Save and apply to** button. The condition is created and applied to the custom response object you made earlier.
11. Click the **Activate** button to deploy your configuration changes. Fastly will now serve your custom HTML error page when required.

§ Enabling automatic gzipping (/guides/basic-configuration/enabling-automatic-gzipping)

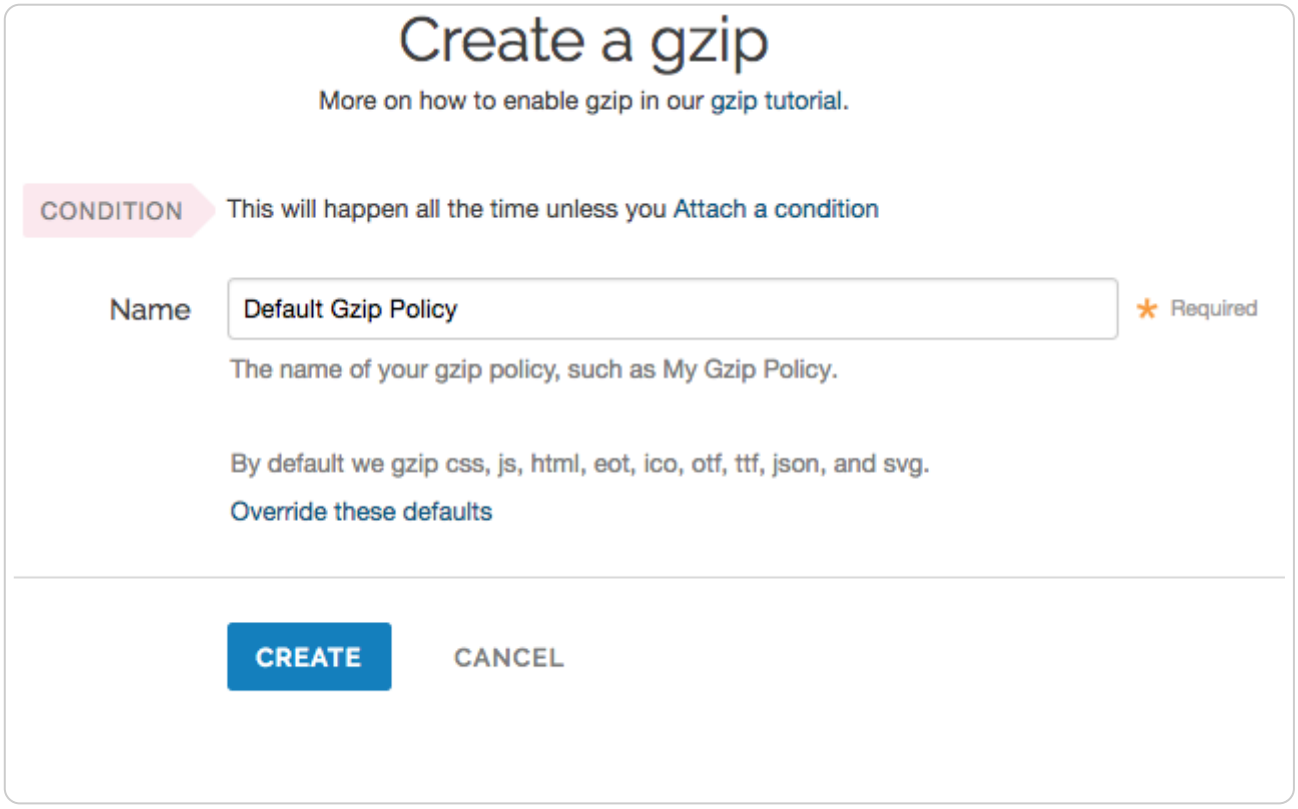
To dynamically gzip cacheable content based on file extension or content-type, follow the steps below.

⚠ **WARNING:** Because dynamic gzipping fetches content from origin, compresses it, and then caches it, this feature doesn't work with our ESI feature (/guides/performance-tuning/esi-use). If you enable gzipping, Fastly will stop processing ESI language elements.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create gzip** button. The Create a gzip page appears.

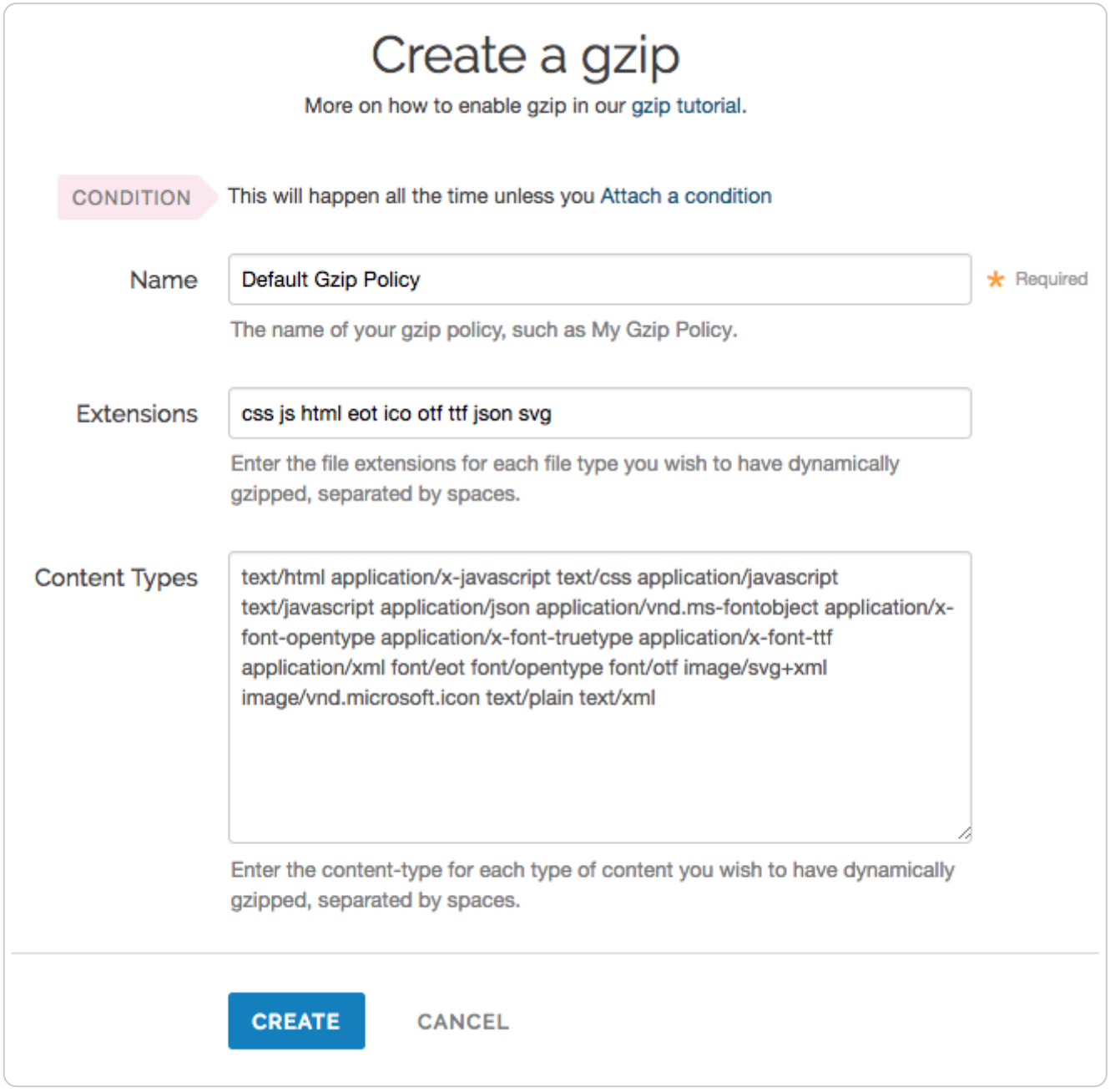


6. Accept the default gzip policy by clicking **Create** button. The Default Gzip policy appears.
7. Click the **Create** button. The new gzip policy appears.
8. Click the **Activate** button to deploy your configuration changes.

Override the default gzip rule

If you want to override the default gzip rule, follow the steps below:

1. Click the **Create gzip** button. The Create a gzip page appears.
2. Click the **Override these defaults** link on the page. The additional gzip fields appear.



3. Fill out the **Create a gzip** fields as follows:
   - In the **Name** field, type an arbitrary name for your new gzip rule.

- In the **Extensions** field, type the file extension for each file type to be dynamically gzipped, separated by spaces. Only type the three- or four-letter string representing the file extension. We recommend setting the Extensions to `css js html eot ico otf ttf json`.
- In the **Content Types** field, type the content-type for each type of content you wish to have dynamically gzipped, separated by spaces. Do not use regular expressions. We recommended setting the Content Types to the following:

```
text/html application/x-javascript text/css application/javascript text/javascript applica
tion/json application/vnd.ms-fontobject application/x-font-opentype application/x-font-tru
etype application/x-font-ttf application/xml font/eot font/opentype font/otf image/svg+xml
image/vnd.microsoft.icon text/plain text/xml
```

4. Click the **Create** button. The new gzip policy appears.
5. Click the **Activate** button to deploy your configuration changes.

Automatic normalization

Because GZip is one of the most common reasons to vary output based on a request header, Fastly will normalize the value of `Accept-Encoding` on incoming requests. The modified header will be set to a single encoding type, or none, and will reflect the best compression scheme supported by the browser. This includes removing Accept-Encoding values in requests from browsers that advertise support for GZip but whose implementation is broken, such as IE6.

Specifically, we run the following steps on inbound requests:

1. If the `User-Agent` matches a pattern for browsers that have problems with compressed responses, remove the `Accept-Encoding` header
2. Else if the `Accept-Encoding` header includes the string "gzip", set the entire value to the string "gzip"
3. Else if the `Accept-Encoding` header includes the string "deflate", set the entire value to the string "deflate"
4. Else remove the `Accept-Encoding` header

Where this normalization process has changed the header value, the original value is made available in the custom header `Fastly-Orig-Accept-Encoding`.

If a user agent advertises support for brotli, currently we will normalize this to gzip because we do not support brotli encoding at the edge. However, if you are doing brotli encoding at your origin server, you may want to modify our normalization algorithm (https://community.fastly.com/t/brotli-compression-support/578/6).

§ Health check frequency (/guides/basic-configuration/health-check-frequency)

Fastly performs health checks (/guides/basic-configuration/health-checks-tutorial) on your origin server based on the Check frequency setting you select in the Create a new health check page. The Check frequency setting you select specifies approximately how many requests per minute Fastly POPs (https://www.fastly.com/network-map) are checked to see if they pass. There is roughly one health check per Fastly POP per period. Any checks that pass will be reported as "healthy."

To determine origin health, you can configure health checks with the following frequency options:

- **Low** - One request every minute from each datacenter, where "healthy" means 1 out of 2 must pass.
- **Medium** - One request every 15 seconds from each datacenter, where "healthy" means 3 out of 5 must pass.
- **High** - One request every 2 seconds from each datacenter, where "healthy" means 7 out of 10 must pass.
- **Custom** - A custom frequency and request health pass levels

To set a custom frequency, select **Custom**. The following controls appear:

- **Threshold & Window** - The number of successes per total number health checks. For example, specifying `1/2` means 1 out of 2 checks must pass to be reported as healthy.
- **Initial** - The number of requests to assume as passing on deploy.
- **Interval & Timeout (ms)** - Interval represents the period of time for the requests to run. Timeout represents the wait time until request is considered failed. Both times are specified in milliseconds.

§ Health checks tutorial (/guides/basic-configuration/health-checks-tutorial)

In this tutorial we show you how to create health checks that periodically contacts your origin servers to make sure they're still working.

We assume that you're already accustomed to editing and deploying configurations using the web interface (https://manage.fastly.com). If you're not familiar with basic editing using the web interface, see our Help Guides (/guides/) to learn more before you continue.

Overview

Health checks, while simple in principle, are more involved than most other configuration objects. Let's run through the options you have when creating a health check so you have a place to start when working through the tutorial.

- **Name** - A human-readable identifier for the health check (e.g., `West Coast Origin Check`).
- **Request** - An HTTP method and path to visit on your origin servers when performing the check. Use a unique path, for example, use `/website-healthcheck.txt`, not `/` or `/healthcheck`.
- **Host header** - The HTTP Host Header to set when making the request (e.g. `example.com`).
- **Expected response** - The HTTP status code the origin servers must respond with for the check to pass (usually `200 OK`).
- **Check frequency** - How often the origin server is checked with additional parameters that determine if the origin server is up or down. For more information about the additional parameters, see our guide on Health check frequency (/guides/basic-configuration/health-check-frequency).
- **Threshold & Window** - The number of successes per total number health checks. For example, specifying `1/2` means 1 out of 2 checks must pass to be reported as healthy.
- **Initial** - The number of requests to assume as passing on deploy.
- **Interval & Timeout (ms)** - Interval represents the period of time for the requests to run. Timeout represents the wait time until request is considered failed. Both times are specified in milliseconds.

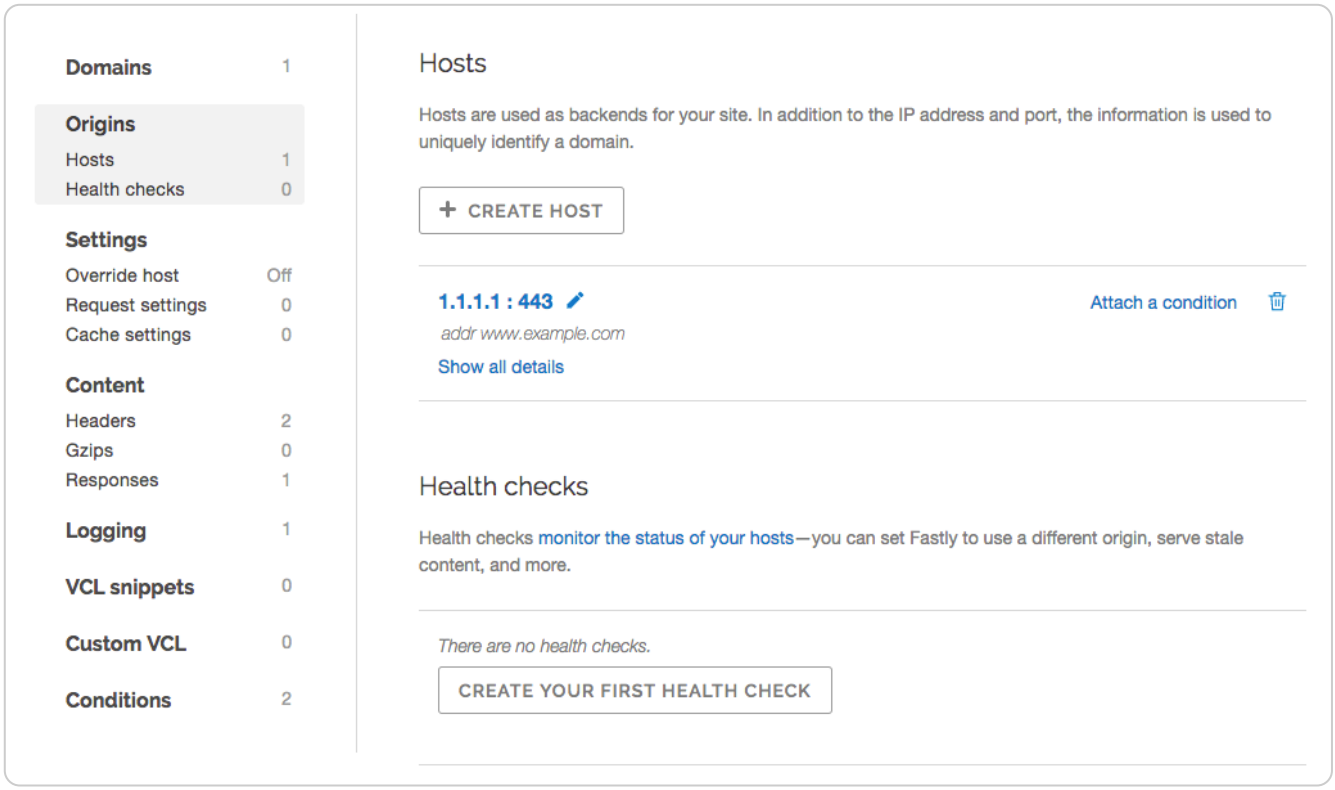
Fastly will periodically check your origin server based on the options chosen. Pay special attention to the HTTP host header. A common mistake is setting the wrong host. If the origin server does not receive a host it expects, it may issue a 301 or 302 redirect causing the health check to fail. Also, Varnish requires the origin server receiving the health check requests to close the connection for each request. If the origin server does not close the connection, health checks will time out and fail.

If an origin server is marked unhealthy due to health checks, Fastly will stop attempting to send requests to it. If all origin servers are marked unhealthy, Fastly will attempt to serve stale (/guides/performance-tuning/serving-stale-content). If no stale object is available, a 503 will be returned to the client.

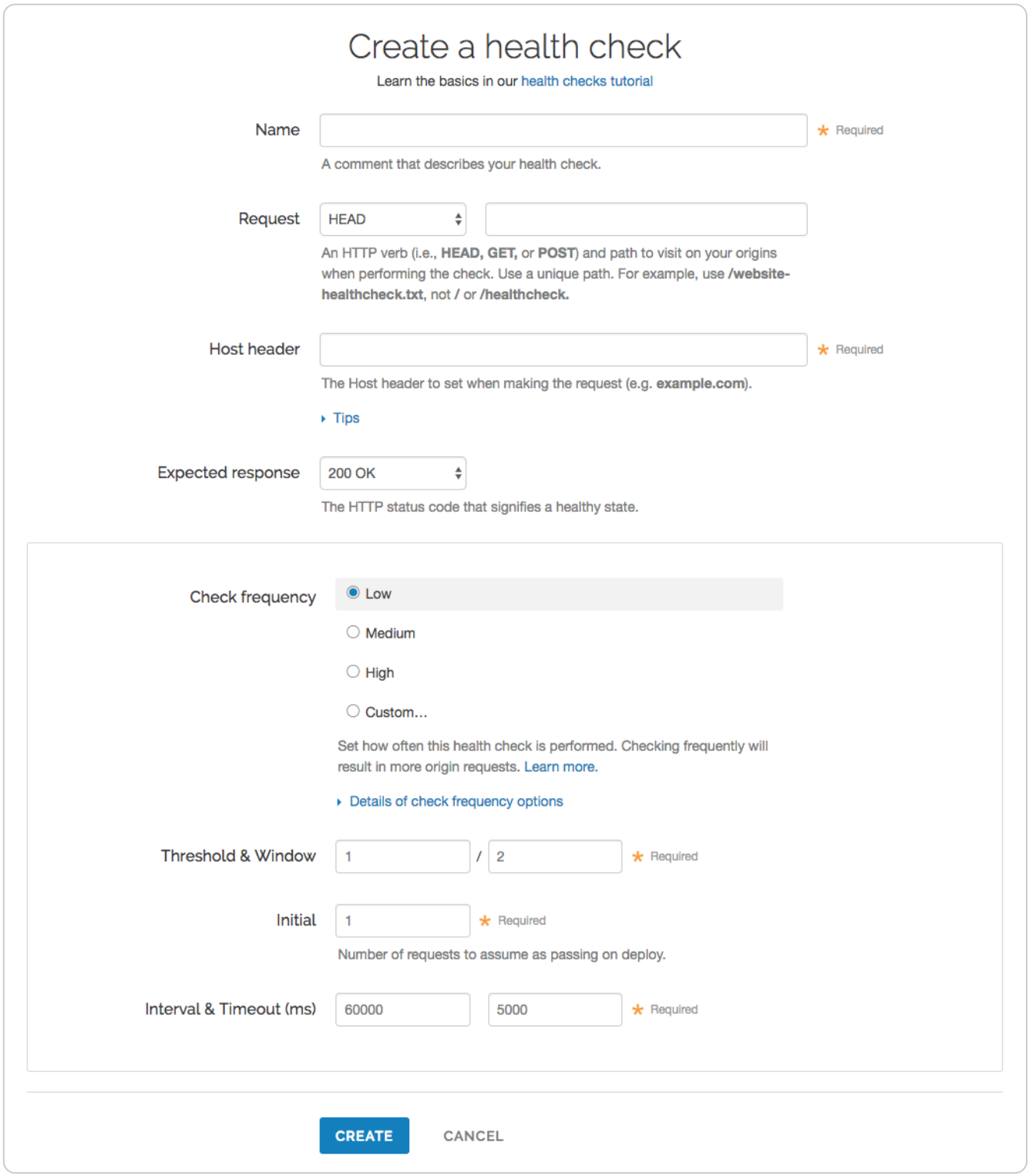
Creating a health check

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.

3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.



5. Click the **Create health check** button. The Create a health check page appears.



6. Fill out the **Create a health check** fields. For more information, review the field descriptions in the Overview section.
7. Click the **Create** button. Your new health check now appears in the list of checks.

Assigning a health check

Health checks do nothing on their own, but they can be added as a special parameter to an origin server in your configuration.

1. Edit one of your existing origin servers by clicking the origin server's name. The Edit this host page appears.
2. From the **Health checks** menu, select the health check you just created.
3. Click **Update**.

Fastly will now use the health check to monitor the selected origin server.

§ How request settings are applied (/guides/basic-configuration/how-request-settings-are-applied)

Requests settings are applied based on the Action you select in the Create a new request setting page. You can choose any one of the following settings:

- **Do nothing now** - Apply the request setting options, but don't force a lookup or a pass action. The request settings are applied as the system continues through the VCL logic.
- **Lookup (in cache)** - Immediately search the cache for content. If the content isn't found (a MISS), then send the request to the origin.
- **Pass (do not cache)** - Immediately send the request to the origin each time and ignore additional request configurations. See our info on understanding the different PASS action behaviors (/guides/vcl/understanding-the-different-pass-action-behaviors) to learn more.

§ Manipulating the X-Forwarded-For header (/guides/basic-configuration/manipulating-the-x-forwarded-for-header)

You can control what happens to the X-Forwarded-For HTTP header via the Create a new request setting page on the Requests settings area of the Settings page. From the X-Forwarded-For menu, select one of the following behaviors:

- **Append** - Appends the client IP to the X-Forwarded-For header.
- **Append All** - Appends the client IP (and edge-cache IP, in case of shielding) to the X-Forwarded-For header. Creates the header if it does not exist yet.

- **Clear** - Clears the X-Forwarded-For header.
- **Leave** - Leaves the X-Forwarded-For header as is, if it is present.
- **Overwrite** - Overwrites the X-Forwarded-For header with just the client IP.

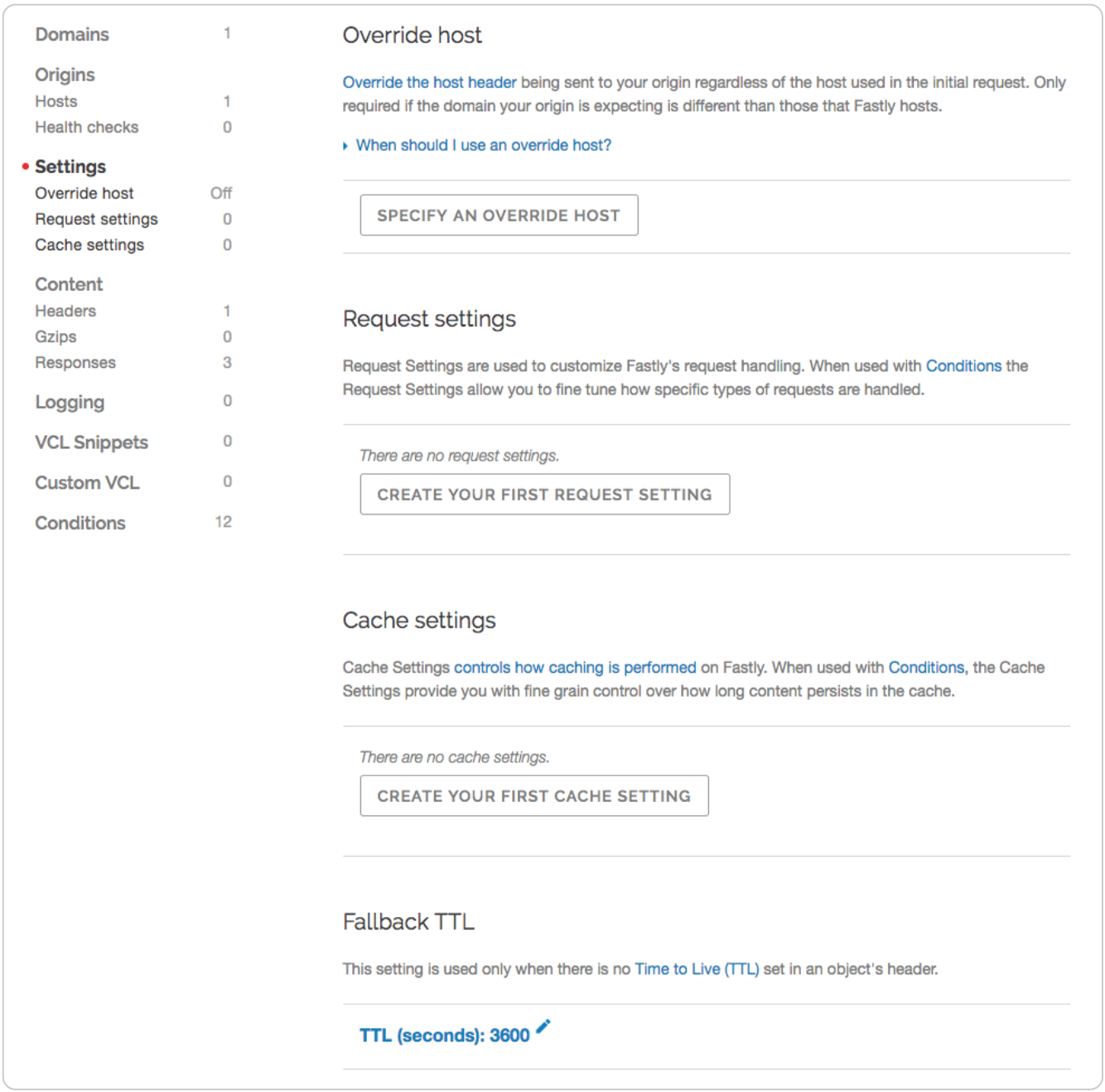
For more information about requests and responses, see our tutorial (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses).

§ Overriding caching defaults based on a backend response (/guides/basic-configuration/overriding-caching-defaults-based-on-a-backend-response)

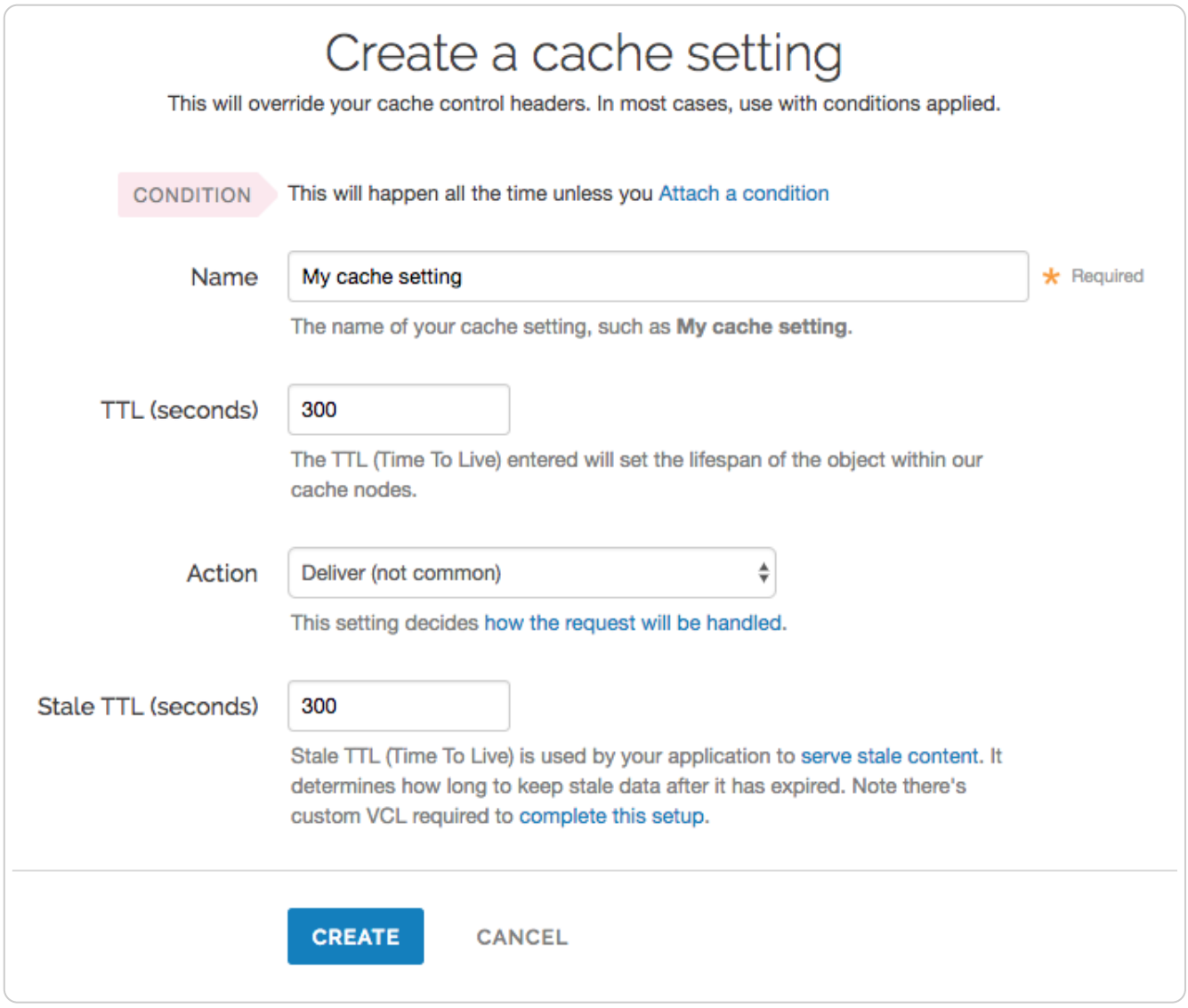
In certain situations you may want to conditionally apply a different caching policy (/guides/performance-tuning/controlling-caching#conditionally-preventing-pages-from-caching) based on a backend response. In this particular case we have backend that on occasion returns 404 errors (e.g., document not found). We don't want those responses to be cached for full caching period of a day but only for 5 minutes. To override default caching we add a cache object and then create conditions for it.

Creating the new Cache Object

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.



5. Click the **Create cache setting** button. The Create a cache setting page appears.



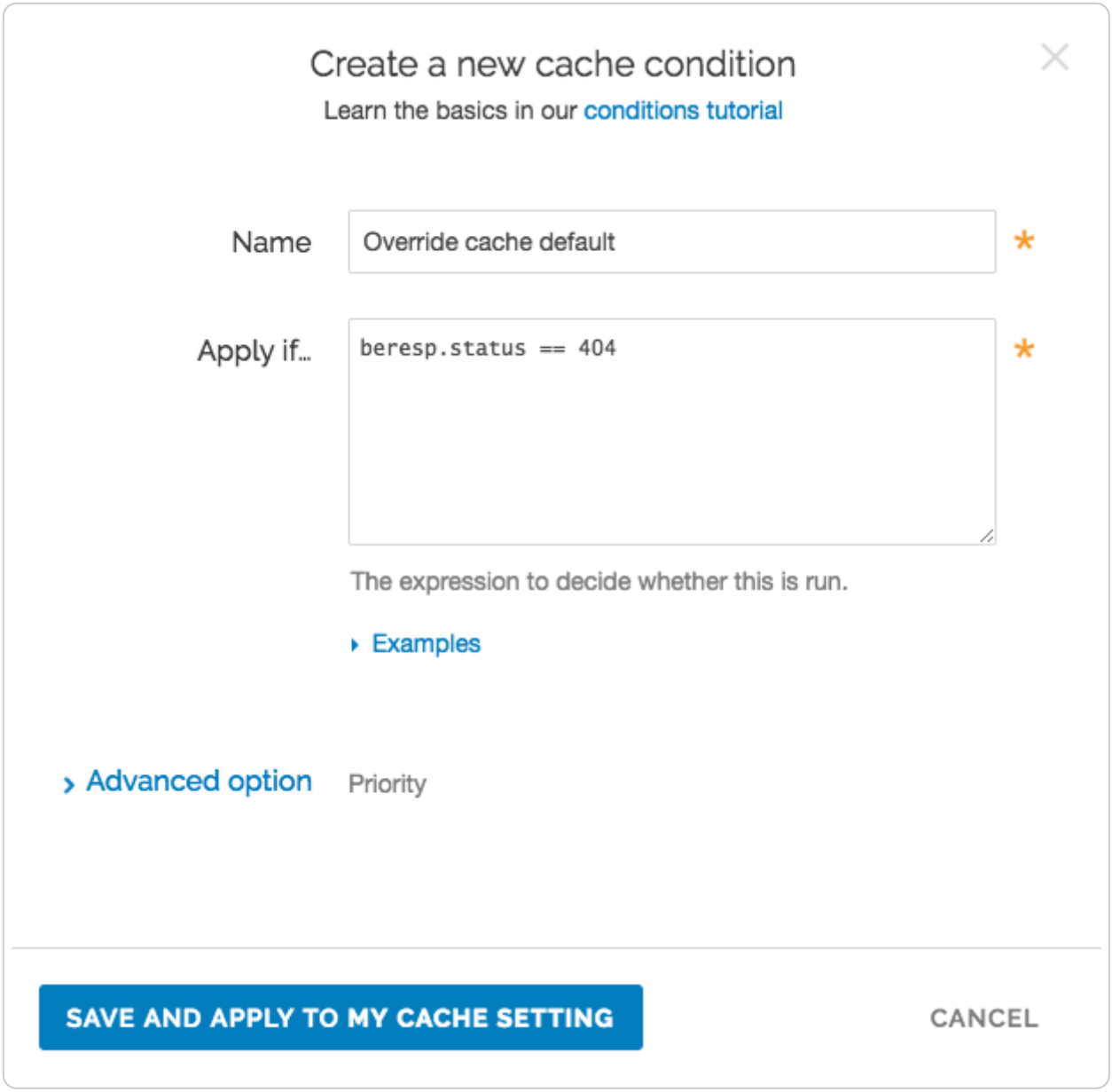
6. Fill out the **Create a cache setting** fields as follows:
   - In the **Name** field, type a descriptive name for the new cache settings.
   - In the **TTL (seconds)** field, type the amount of time, in seconds, to cache the objects (e.g., `300`).
   - From the **Action** menu, select **Deliver**.
   - In the **Stale TTL (seconds)** field, type the amount of time to serve stale or expired responses, in seconds, should the backend become unavailable (e.g., `300`).
7. Click the **Create** button.

Creating an Override Condition for the new Cache Object

Once the object is created, add a condition to it.

1. Click the **Attach a condition** link to the right of the object.

2. Click **Create cache setting** button. The Create a new cache condition window appears.



3. Fill out the **Create a new cache setting** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition. For example, `Override cache default`.
   - In the **Apply if** field, type an appropriate backend response header to specify when the condition will be applied. For example, `beresp.status == 404`.
4. Click the **Save and apply to** button.
5. Click the **Activate** button to deploy your configuration changes.

Other notes

You can use any backend response header in the **Apply if** field to make decisions on caching.

For example, `beresp.http.Content-Type ~ "^text/html"` can be used to specify different caching rules for HTML documents.

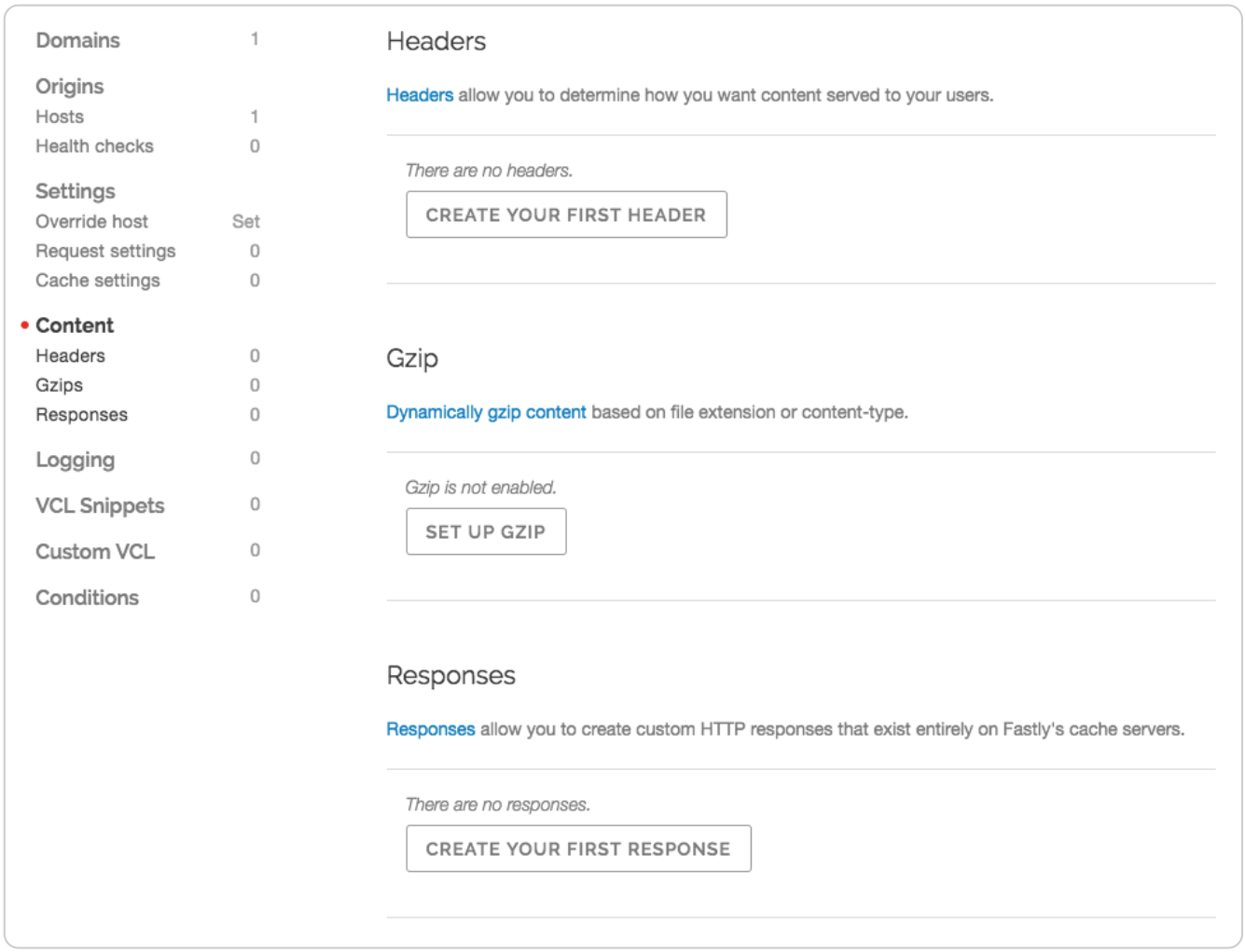
§ Removing headers from backend response (/guides/basic-configuration/removing-headers-from-backend-response)

You can remove headers from any backend response. This may be necessary if your application automatically sets headers. For example, Drupal can set the following Expires and Cache-Control headers to prevent caching:

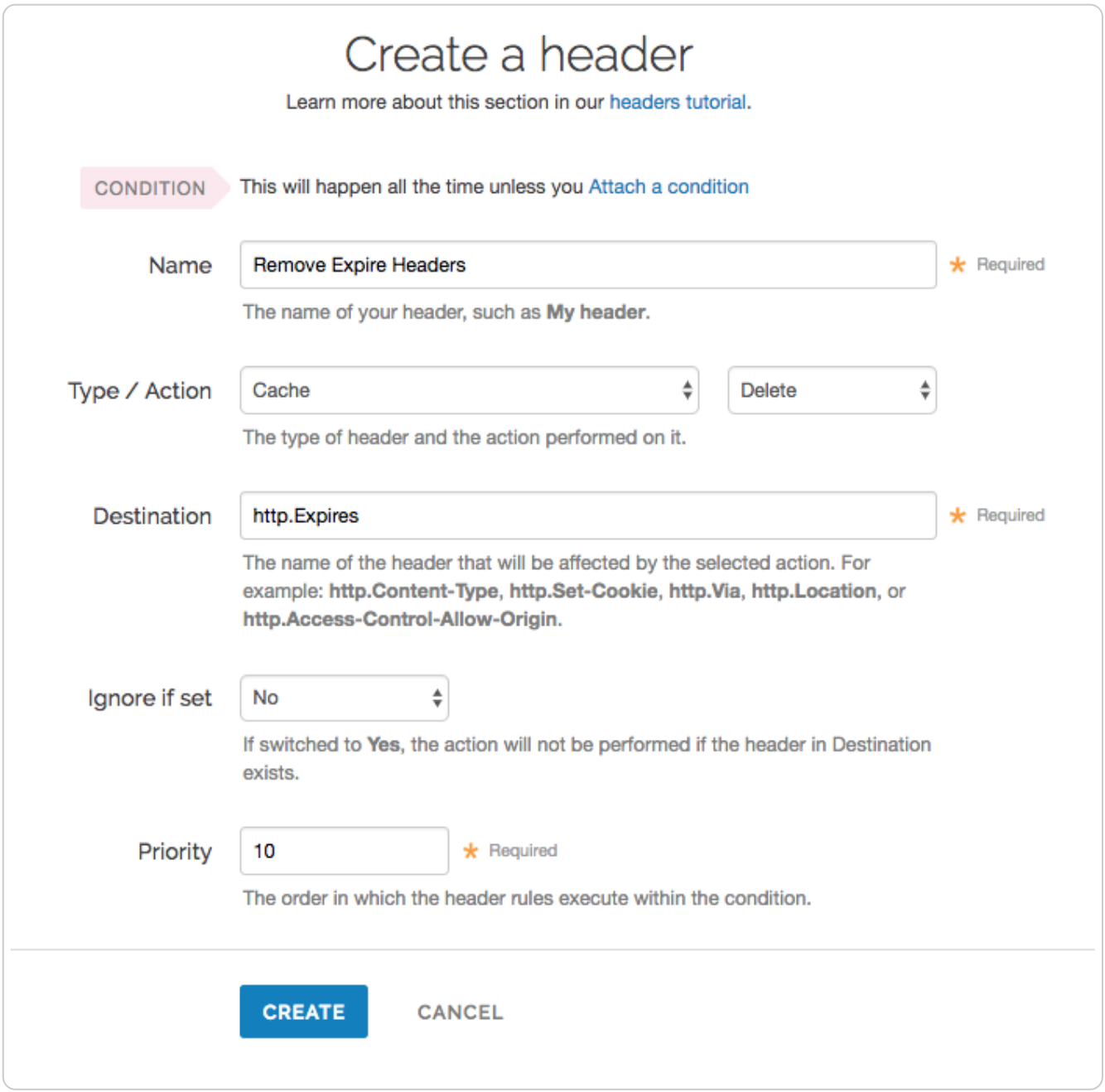
```
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Last-Modified: Wed, 18 Jul 2012 18:52:16 +0000
Cache-Control: no-cache, must-revalidate, post-check=0, pre-check=0
```

To remove a header from the backend response, add a new header as follows:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the header rule (e.g., `Remove Expire Headers`).
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Delete**
   - In the **Destination** field, type the name of the header (e.g., `http.Expires`).
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `10`.
7. Click the **Create** button.
8. Click the **Activate** button to deploy your configuration changes.

★ **TIP:** You may also be interested in our information on setting content type based on file extension (/guides/basic-configuration/setting-content-type-based-on-file-extension).

§ Responses tutorial (/guides/basic-configuration/responses-tutorial)

Fastly allows you to create custom HTTP responses that are served directly from the cache without storing the page on a server. Responses are commonly used to serve small static assets that seldom change and maintenance pages that are served when origins are unavailable. This tutorial shows you how to create your own responses.

🛈 **NOTE:** We assume that you already know how to edit and deploy configurations using the web interface (https://manage.fastly.com/). If you are not familiar with basic editing using the application, see our help guides (/guides/) to learn more.

Overview

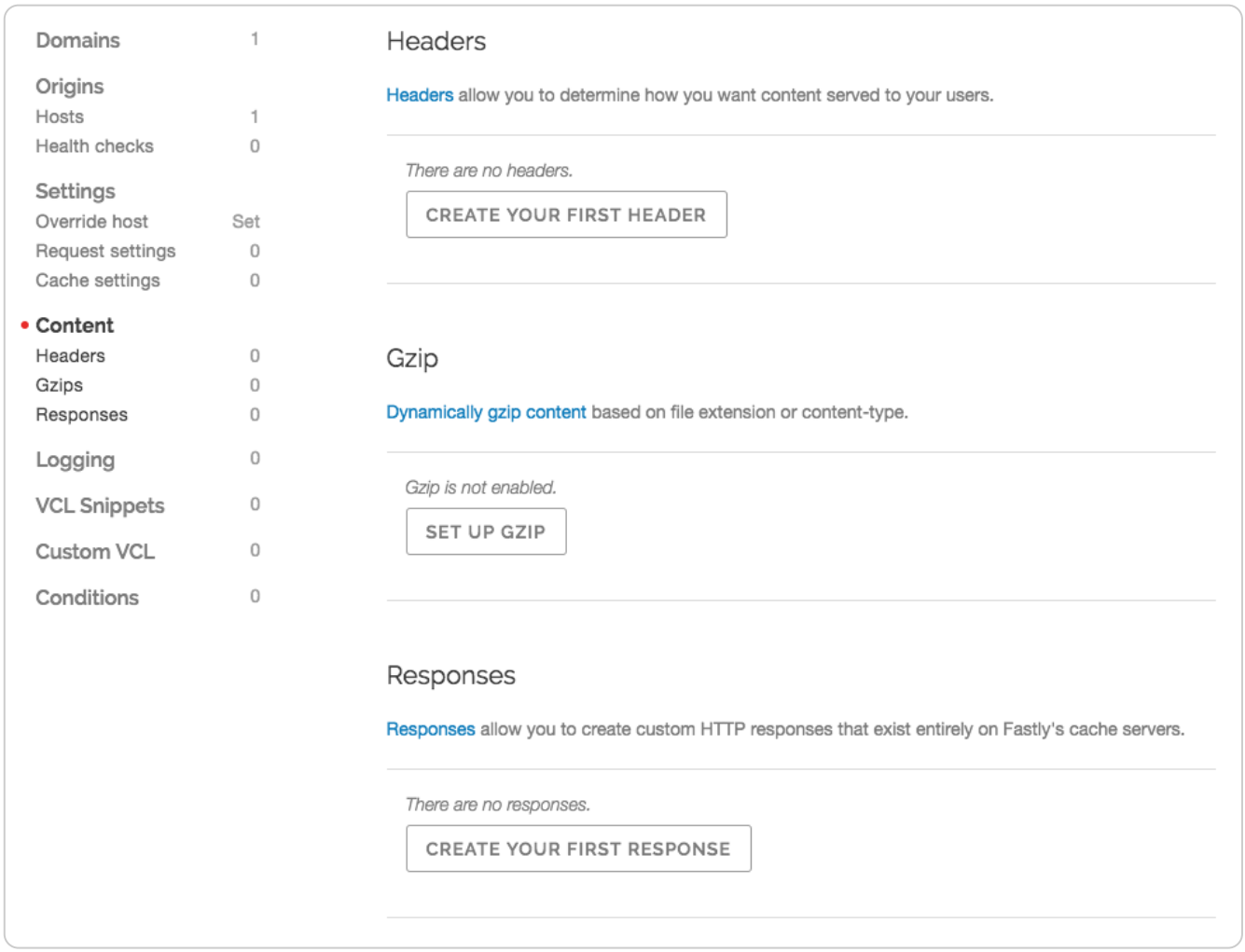
A response has three basic attributes:

- Status - An HTTP status code to include in the header of the response
- Response - The content to be served when delivering the response
- Description - A human readable identifier for the response

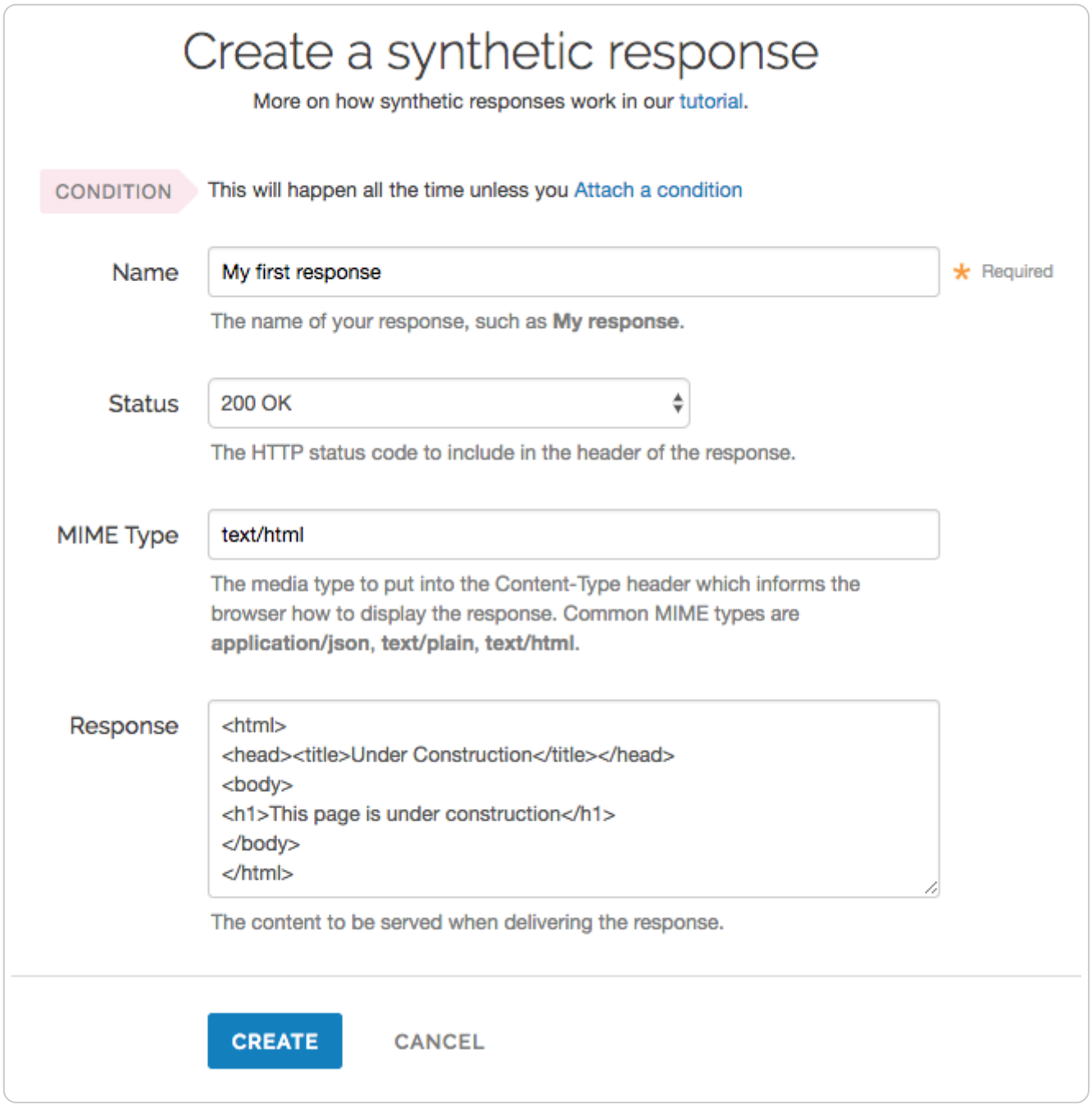
By setting these three attributes and adding a condition to the response, you can very quickly get one up and running on your service. Let's get started!

Creating a response

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create response** button. The Create a synthetic response page appears.

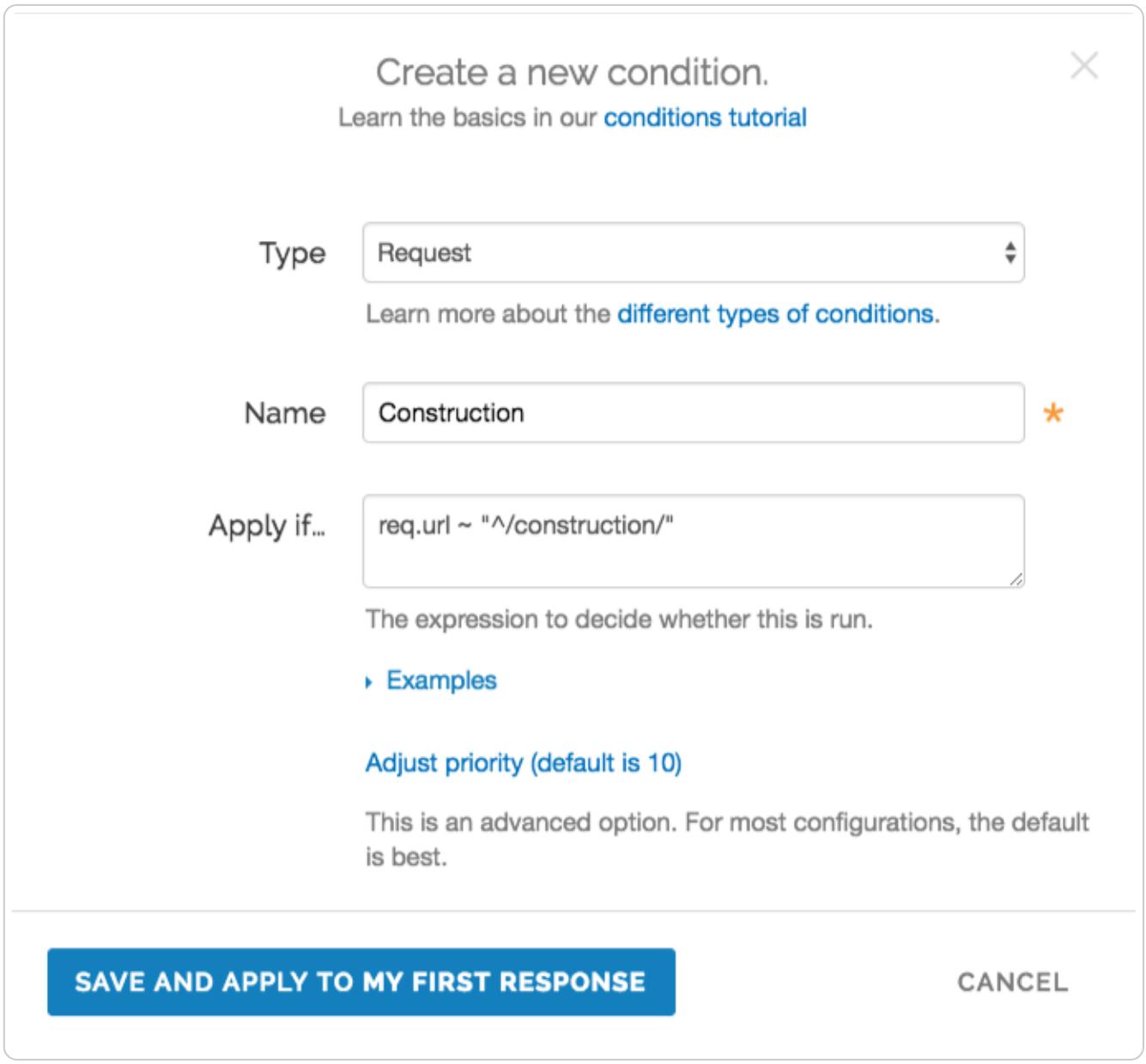


6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type a human-readable name for the response (e.g., `My first response`).
   - From the **Status** menu, select the appropriate status (e.g., `200 OK`).
   - In the **MIME Type** field, type the content type of the response (e.g., `text/html`).
   - In the **Response** field, type the response you want to appear when the conditions are met.
7. Click the **Create** button to create your custom response.

Your new response appears in the list of responses.

Adding conditions

1. Click the **Attach condition** link to the right of the new response. The Create a new condition window appears.



2. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select the type of condition you want to create.
   - In the **Name** field, type a human-readable name for the condition so that it can be easily identified in the future.
   - In the **Apply if** field, type the condition under which the new response occurs. The condition should take the following format: `req.url ~ "^/construction/"` equals the request condition you're creating the response for. The Conditions subcategory (/guides/conditions/) has more detailed information on conditions.

- In the **Priority** field, type a priority if needed. Condition priorities are only needed in "interesting" cases, and can usually be left at the default "10" for all response conditions.

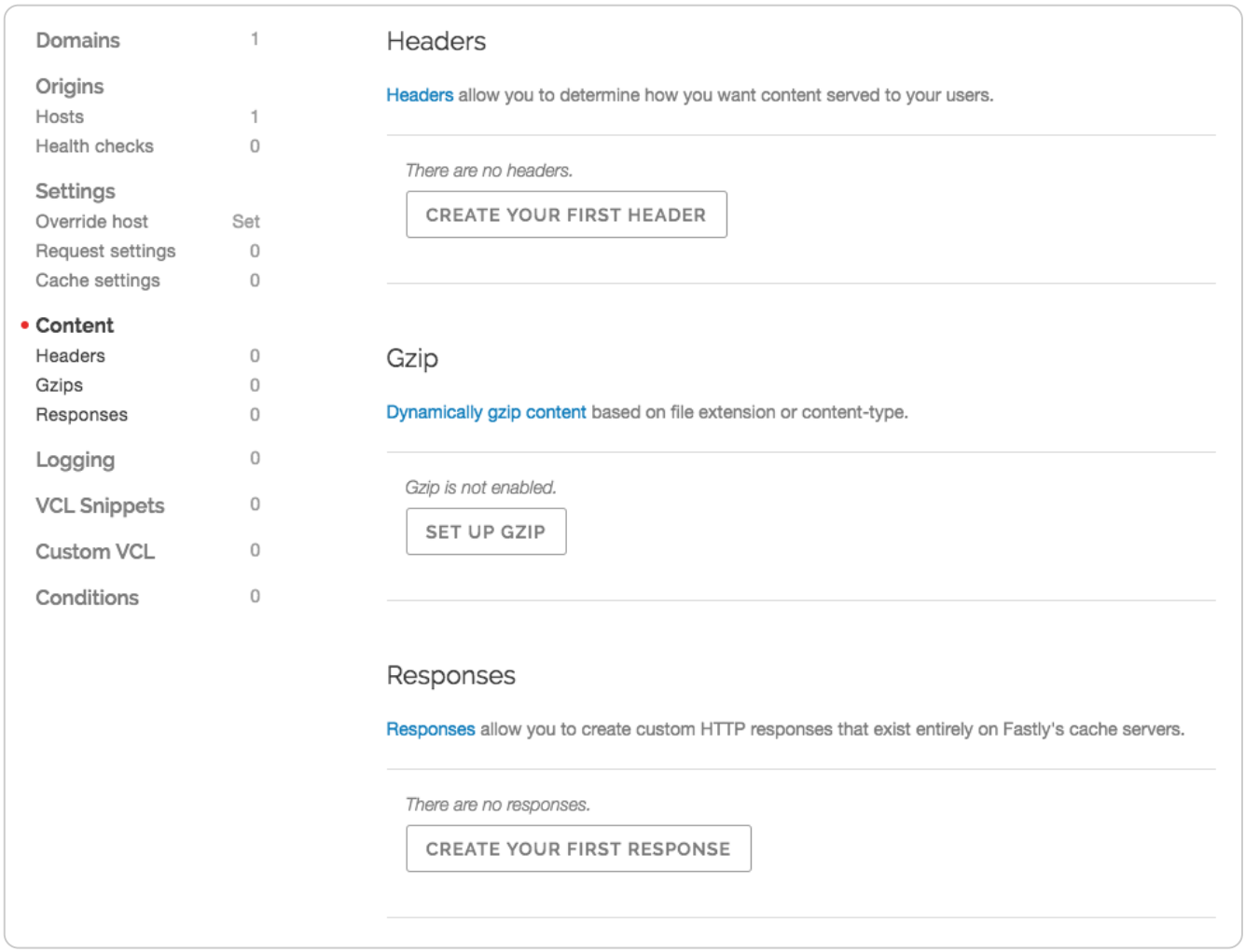
3. Click the **Save and apply to** button.
4. Click the **Activate** button to deploy your configuration changes.

Fastly now serves your custom response page when the condition is met.

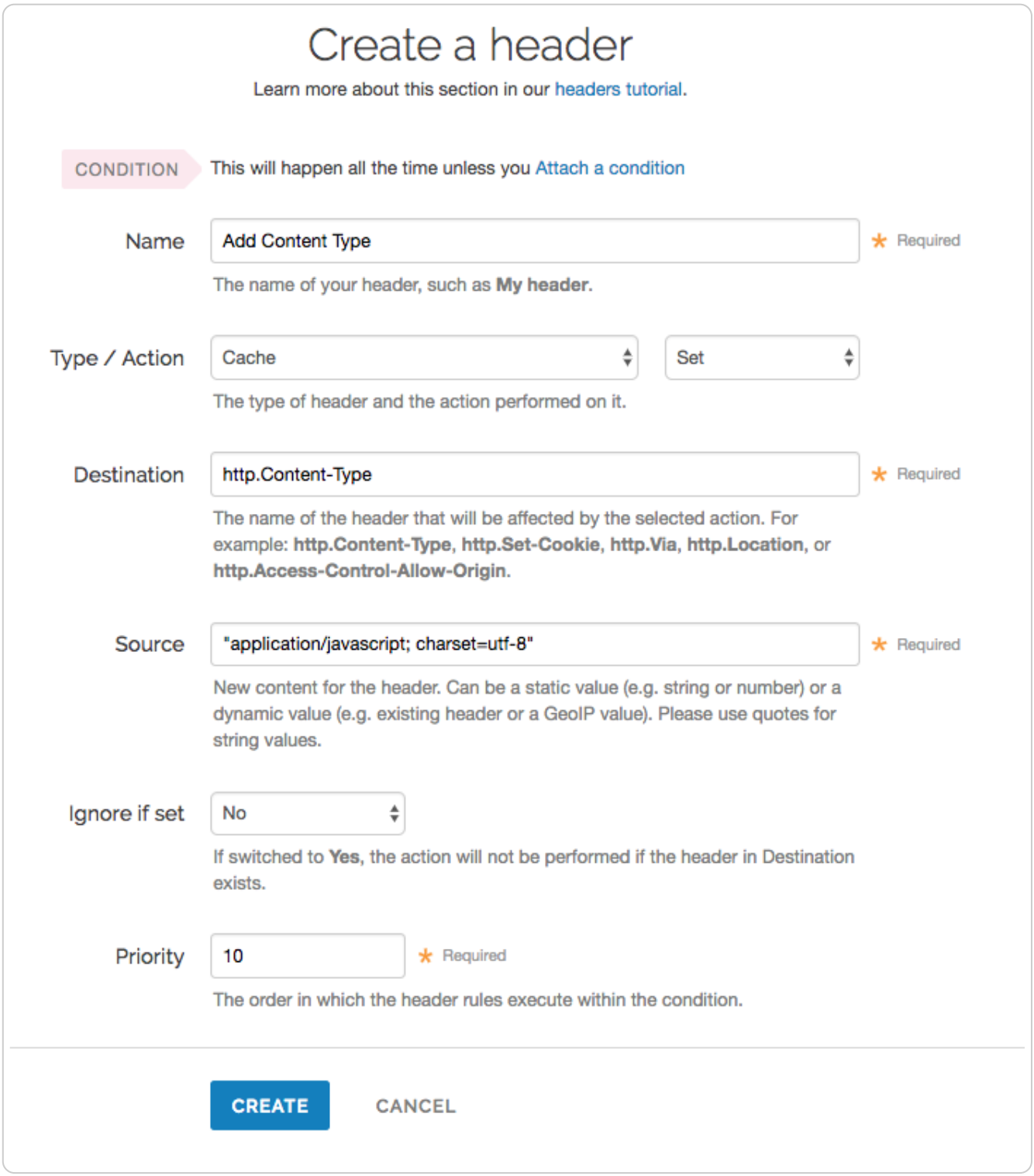
§ Setting Content Type based on file extension (/guides/basic-configuration/setting-content-type-based-on-file-extension)

In some situations you may want to override the content type that a backend returns. To do that you will need to create a new header object and an associated condition.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header page appears.



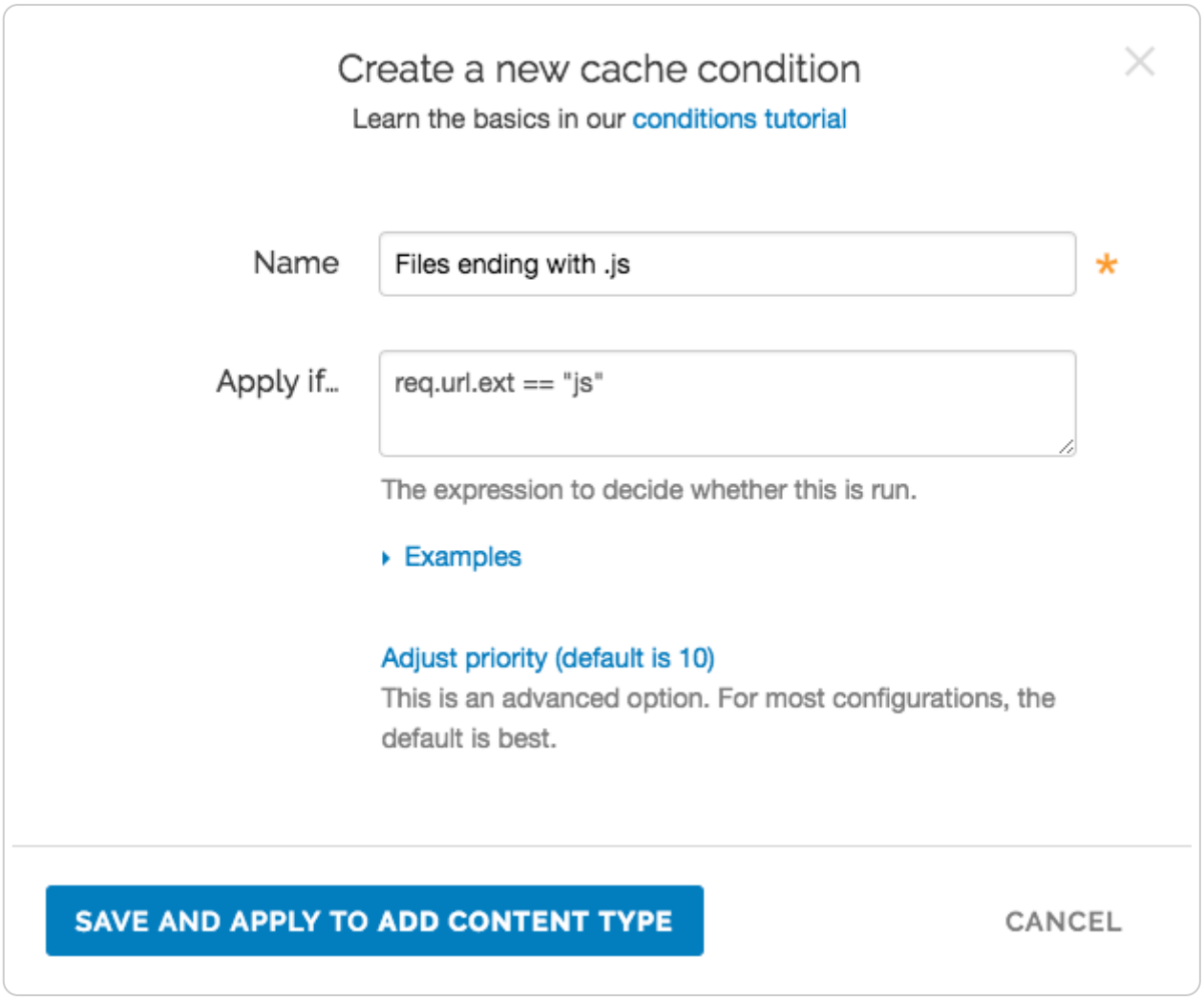
6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type an appropriate name (e.g., `Add Content Type`).
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Content-Type`.
   - In the **Source** field, type the content type you want to match, such as `"application/javascript; charset=utf-8"`.

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button.

Once you have created the header object, apply a condition (/guides/conditions/). Otherwise, that particular object is applied to all requests.

1. Click the **Attach a condition** link to the right of the new header name. The Create a new cache condition window appears.



2. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, type a descriptive name, such as `Files ending with .js`.
   - In the **Apply if** field, type the condition that matches your request, such as `req.url.ext == "js"` (to match the request for files ending in .js).
3. Click the **Save and apply to** button to create the new condition.
4. Click the **Activate** button to deploy your configuration changes.

★ **TIP:** You may also be interested in our guide to Removing headers from backend response (/guides/basic-configuration/removing-headers-from-backend-response).

§ Specifying an override host (/guides/basic-configuration/specifying-an-override-host)

If you want to rewrite the host header being sent to your origin regardless of the host used in the initial request, specify an override host. Use this if you have multiple domains tied to a service and want them all served by the same origin, or if the domain your origin is expecting is different than one specified in your Fastly service. You most likely won't need to use this feature.

You can override the host header being sent to your origin by specifying the domain name (/guides/basic-configuration/working-with-domains) of your override host on the **Settings** page for a specific service.

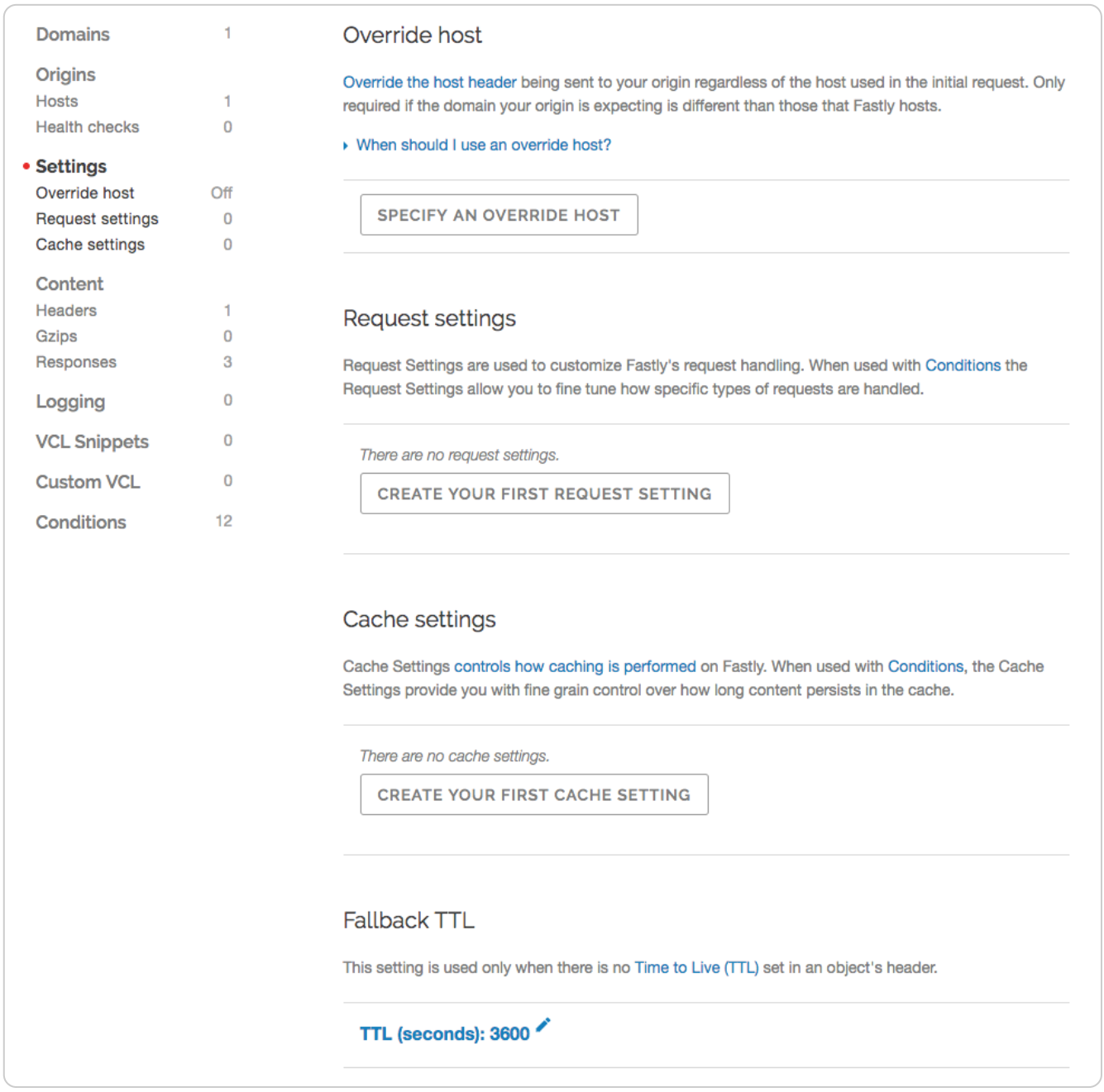
Here are some examples of when to use an override host:

- When using backends such as Amazon S3 (/guides/integrations/amazon-s3), Google Cloud Storage (/guides/integrations/google-cloud-storage#setting-the-default-host-for-your-service-to-your-gcs-bucket), or Heroku (https://devcenter.heroku.com/articles/fastly), you want to ensure you use the proper host header so these providers know how to route requests directly to your content. Each provider uses the host header to associate requests with your account's storage location. For example, if you set up your origin using Amazon S3, you send the name of your S3 bucket as your host header. Amazon is set up so that it only accepts host headers that have the same name as the bucket hosting your content. A request to `your-domain.com` must be re-written to `<your-bucket>.s3.amazonaws.com`, or else the request is denied.
- You have a service that contains three sites: `www.abc.com`, `www.myexample.com`, and `www.mysite.com` and you have one origin. You can have the same origin respond to each domain by overriding the host header to one accepted by your origin, for example, `origin.example.com`. The result will be that a request to `www.abc.com`, `www.myexample.com`, or `www.mysite.com` returns content from `origin.example.com`.

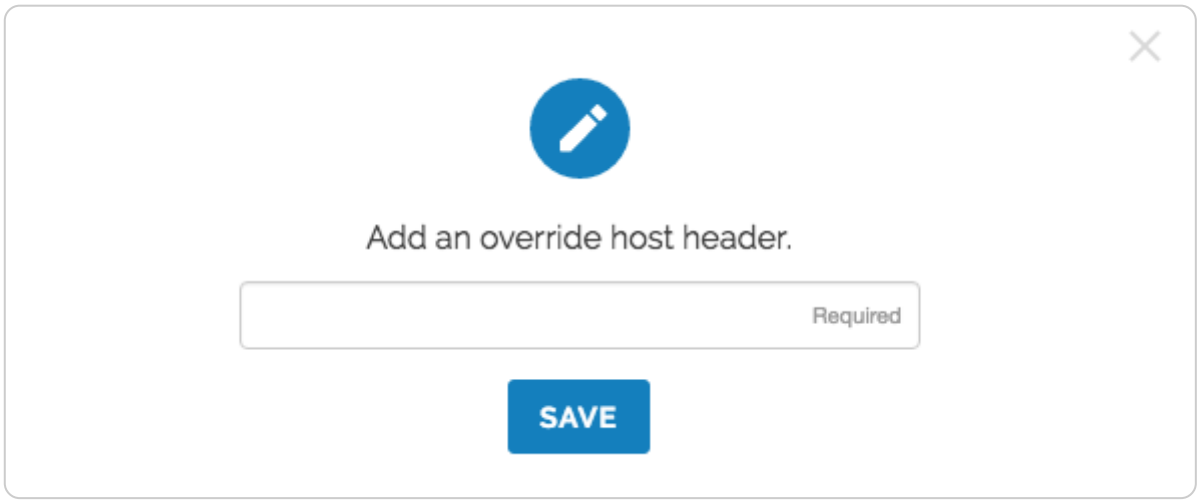
Override a host

1. Log in to the Fastly web interface and click the **Configure** link.

2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.



5. Click the **Specify an Override Host** button. The Add an override host header window appears.



6. Type the hostname of your override host based on the origin you are using:
   - If you are using Amazon S3 (/guides/integrations/amazon-s3) as your origin, type `<yourbucket>.s3.amazonaws.com`.
   - If you are using Google Cloud Storage (/guides/integrations/google-cloud-storage#setting-the-default-host-for-your-service-to-your-gcs-bucket) as your origin, type `<your bucket name>.storage.googleapis.com`.
7. Click the **Save** button. The override host header appears in the Override host section.
8. Click the **Activate** button to deploy your configuration changes.

Caveats about using the override host

There are situations when you may not want to use an override host:

- **Using multiple origins.** When you specify a host override, you're specifying what hostname is actually sent to your origin. If you have a service with two different origins and each origin requires a different hostname, specifying a host override for all requests results in one origin not returning valid responses. If you specify a default hostname that matches only one of the origins, then no content is returned from the other origin requests.

  > ℹ **NOTE:** If you want to serve content from multiple backends, you should conditionally route to them. Refer to Routing assets to different origins (/guides/performance-tuning/routing-assets-to-different-origins) for more information.

- **Shielding is enabled.** If you enable a host override along with shielding and the specified override host doesn't match to a domain within the service, the shield won't route the request properly and an error of 500 is expected. Refer to Shielding (/guides/performance-tuning/shielding#caveats-of-shielding) for more information.

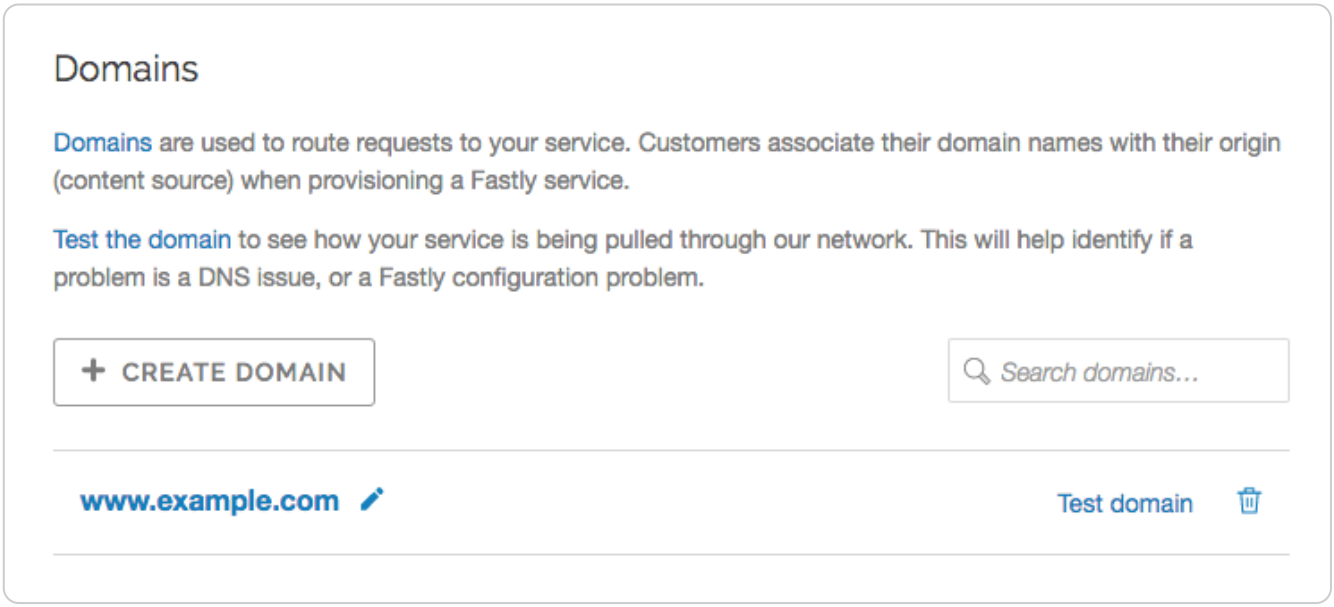
§ Testing setup before changing domains (/guides/basic-configuration/testing-setup-before-changing-domains)

After you deploy your service, but before you change your DNS entries (/guides/basic-setup/adding-cname-records) to send your domain (/guides/basic-configuration/working-with-domains) to our servers, you can check to see how your service is pulled through our network. Testing your domain can help you identify DNS issues or problems with your Fastly configuration.

Using the web interface

To use the web interface to test your domain on Fastly before you make a final CNAME (/guides/basic-setup/adding-cname-records) change, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.



4. Click the **Test domain** link next to the domain you want to test.
5. Verify that your website appears in a new tab in your web browser.

Using command line utilities

To use command line utilities to test your domain on Fastly before you make a final CNAME (/guides/basic-setup/adding-cname-records) change, you would:

- find the IP address of a Fastly pop

- add a domain host entry to your hosts file
- test the domain in a web browser

Determining the IP address of a Fastly POP

Use the nslookup or dig command to determine the IP address of a Fastly POP.

> ★ **TIP:** For non-TLS requests, use `nonssl.global.fastly.net`. For TLS requests, use the custom TLS CNAME record (/guides/basic-setup/adding-cname-records#tls-enabled-hostnames) provided by Fastly support. For more information about the Fastly TLS service, see our guide on TLS service options (/guides/detailed-product-descriptions/tls-service-options).

For example, running nslookup for `nonssl.global.fastly.net` returns:

```
$ nslookup nonssl.global.fastly.net
Server:         185.121.177.177
Address:        185.121.177.177#53

Non-authoritative answer:
Name:   nonssl.global.fastly.net
Address: 151.101.56.204
```

Find the IP address at the bottom of the nslookup response. In this example, it's `151.101.56.204`.

Alternatively, running dig for `nonssl.global.fastly.net` returns:

```
$ dig nonssl.global.fastly.net

; <<>> DiG 9.8.3-P1 <<>> nonssl.global.fastly.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35146
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;nonssl.global.fastly.net.      IN      A

;; ANSWER SECTION:
nonssl.global.fastly.net. 30    IN      A       151.101.56.204
```

The IP address (A record) is in the `ANSWER SECTION` of the dig results: `151.101.56.204`.

Modifying your hosts file

You can temporarily add a static IP address and domain host entry to the hosts file on your computer. For example, if the domain you are testing is `www.example.com` and one of the IP addresses returned by nslookup or a dig command is `151.101.56.204`, you would add this entry to the file:

```
151.101.56.204 www.example.com
```

and save the changes.

> ★ **TIP:** On machines running Mac OSX or Linux, your hosts file is `/etc/hosts`. On Windows-based machines, it's `C:\Windows\System32\Drivers\etc\hosts`.

Testing your domain

Test your domain to see how Fastly pulls it through our network by restarting your browser if it's already running, and then typing your domain in the address field. You should now see the updated domain in the address field indicating requests are being sent to the Fastly POP.

Alternatively, you can test the domain using a ping command to verify that your domain is being served by a Fastly POP address. In this case, `ping www.example.com` would display the Fastly POP address `151.101.56.204`.

Be sure to remove the host entry from your hosts file after you make CNAME changes to point your domain to Fastly.

---

§ Using Fastly with apex domains (/guides/basic-configuration/using-fastly-with-apex-domains)

Some customers use only their second level or apex domain (e.g., `example.com` rather than `www.example.com`) as their canonical domain. Due to limitations in the DNS specification, we don't recommend placing a CNAME record at the apex domain or using the CNAME Flattening (e.g., ALIAS or ANAME) features offered by some DNS providers. Instead, we offer Anycast IP addresses for content that must be hosted on a second-level or apex domain.

Where problems exist and why

The DNS instructions in RFC1034 (https://www.ietf.org/rfc/rfc1034.txt) (section 3.6.2) state that, if a CNAME record is present at a node, no other data should be present. This ensures the data for a canonical name and its aliases cannot be different. Because an apex domain requires NS records and usually other records like MX to make it work, setting a CNAME at the apex would break the "no other data should be present" rule.

In general, the problem with apex domains happens when they fail to redirect to their `www` equivalents (`example.com` points nowhere instead of pointing to `www.example.com`). A couple of workaround options exist:

- Only use Fastly for API or AJAX calls, images, and other static assets (e.g., serve `example.com` yourself and CNAME to Fastly for assets at `assets.example.com`)
- Redirect from the apex domain to the version proxied by Fastly (e.g., redirect any requests for `example.com` to `www.example.com`)

Neither of these workarounds, however, are ideal.

Anycast option

Fastly can provide Anycast (https://en.wikipedia.org/wiki/Anycast) IP addresses for content that must be hosted on a second-level or apex domain. We do not charge extra for this service, however, you must be using one of Fastly's paid plans (/guides/account-types-and-billing/accounts-and-pricing-plans) (with or without a contract) in order to take advantage of it. Also, if you do use Anycast IP addresses, using these IPs may result in higher rates (/guides/account-types-and-billing/how-we-calculate-your-bill).

Our Anycast option allows you to add A records that point your apex domain at Fastly. If the Fastly Anycast IP addresses change we will notify you at status.fastly.com (https://status.fastly.com/). Because Anycast doesn't give us as much flexibility in routing your requests, this option may not be as performant as our CNAME-based system (/guides/basic-setup/adding-cname-records). We recommend you use our CNAME-based system for as much content as possible, particularly for large files or streaming video.

For questions about Anycast and second-level or apex domains, contact support@fastly.com (mailto:support@fastly.com).

---

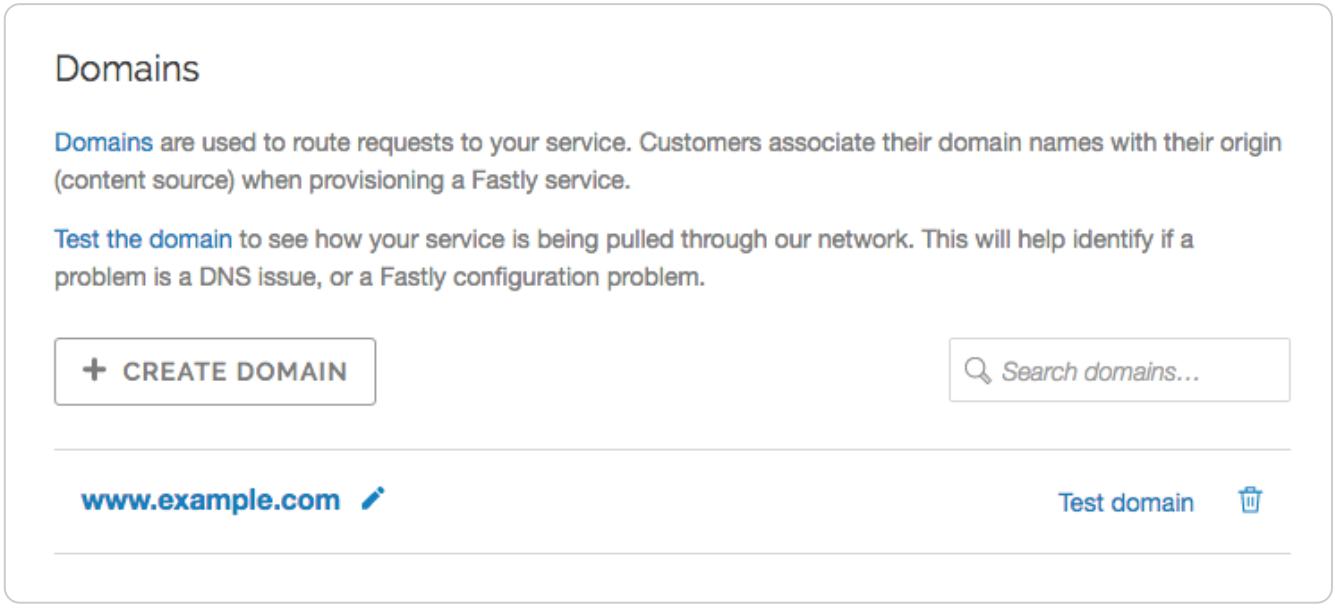
§ Working with domains (/guides/basic-configuration/working-with-domains)

Domains are used to route requests to your service. You associate domain names with your origin when provisioning a Fastly service, and you can add, edit, or remove domains from your service at any time.

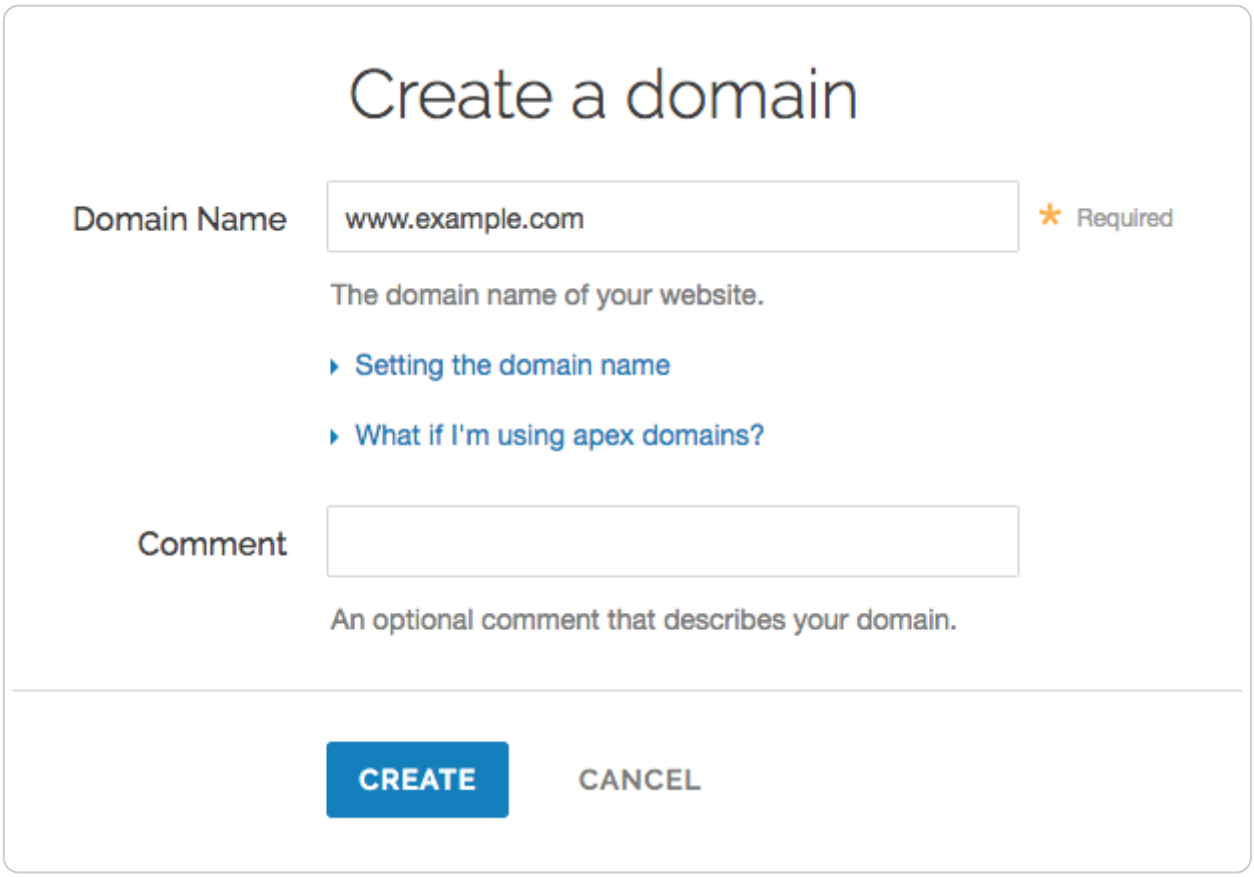
Creating a domain

Follow the steps below to add a domain to your service:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.



4. Click the **Create domain** button. The Create a domain page appears.



5. Fill out the **Create a domain** fields as follows:
   - In the **Domain Name** field, type the domain name. The domain name is used to properly route requests to your website, and ensures that others cannot serve requests to that domain. For example, you could enter `www.example.com`, `blog.example.com`, or even use wildcards such as `*.example.com`.

   > ★ **TIP:** Due to limitations in the DNS specification, Fastly doesn't recommend using apex or second level domains (/guides/basic-configuration/using-fastly-with-apex-domains). An example of an apex domain is `example.com` rather than `www.example.com`.

   - In the **Comment** field, optionally type a comment that describes the domain.
6. Click the **Create** button. Your new domain appears in the list of domains.
7. If you haven't already, add CNAME DNS records (/guides/basic-setup/adding-cname-records) for your domain name to begin routing client traffic through Fastly services instead of directly to your origin.
8. Click the **Activate** button to deploy your configuration changes.

Using the API

You can use Fastly's API (/api/config#domain) to programmatically add domains to your service. To add a domain to your service, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/vers
ion/<version_id>/domain -d 'name=www.example.com'
```

The response will look like this:

```
{
  "comment": "",
  "name": "www.example.com",
  "service_id": "<service_id>",
  "version": <version_id>
}
```

Domain creation limits

We set a limit (/guides/debugging/common-service-and-domain-errors) on the number of domains you can create per service by default. However, if you email support@fastly.com (mailto:support@fastly.com), we may be able to adjust this number for you by working with you to set up and fine-tune domain handling in your service.

Testing a domain

You can test a domain (/guides/basic-configuration/testing-setup-before-changing-domains) to see how your service is being pulled through our network. This can help you identify DNS issues and problems with your Fastly configuration. See our testing guide (/guides/basic-configuration/testing-setup-before-changing-domains) for instructions.

Deleting a domain

Follow the steps below to delete a domain from your service:

1. On the Domains page, click the trash icon next to the domain you want to delete.
2. Click the **Confirm and delete** button to confirm you want to delete your domain.
3. Click the **Activate** button to deploy your configuration changes.

IMPORTANT: To minimize the risk of unauthorized use of your domains, we strongly recommend modifying or deleting any DNS CNAME records (/guides/basic-setup/adding-cname-records) pointing to the Fastly hostname associated with the deleted domain. Follow the instructions on your DNS provider's website.

---

§ About the web interface controls (/guides/basic-concepts/about-the-web-interface-controls)

In addition to being accessible via Fastly's application programming interface (API) (/guides/detailed-product-descriptions/about-fastlys-application-programming-interface), Fastly services can also be accessed via a web interface for users with the appropriate access permissions.

Access to Fastly's web interface controls

Access to Fastly's web interface controls requires you to sign up for a Fastly account. Creating an account is free. Once you've created an account, you can navigate to the controls via the Fastly login page at https://manage.fastly.com (https://manage.fastly.com), either directly using any standard web browser or by clicking the Login link at the top right of almost all pages at the Fastly website (https://www.fastly.com).

Once logged in to a Fastly account, the web interface controls appear as appropriate based on the roles and permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions) assigned to you.



The default control groups appear as follows from left to right across the top of the interface:

- the All services page
- the Dashboard page
- the Stats page
- the Configure controls
- the user menu

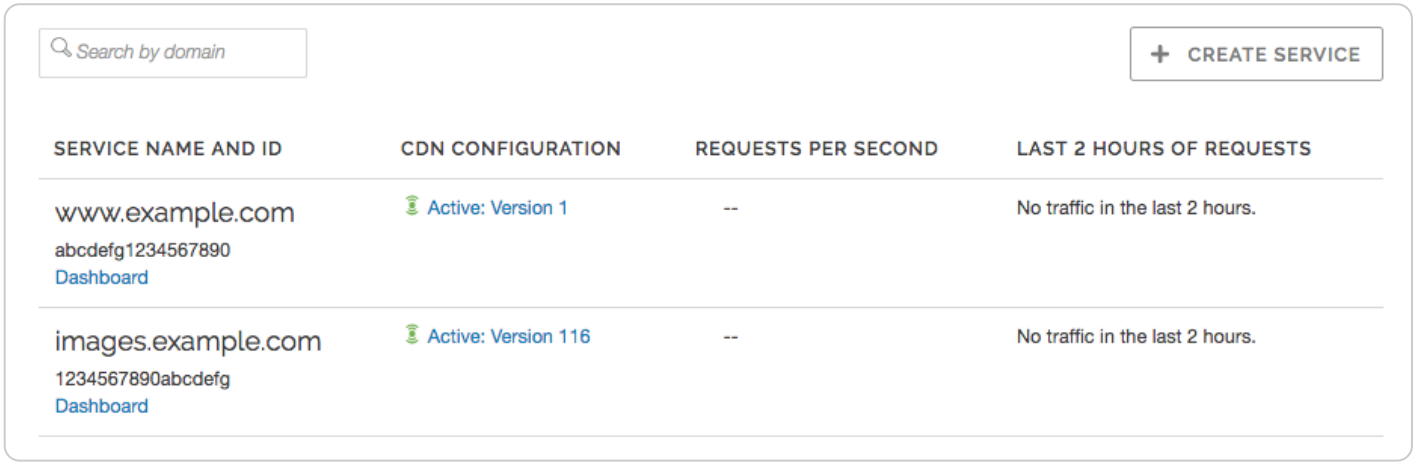
NOTE: Not all Fastly service features are enabled by default (e.g., custom VCL (/guides/vcl/uploading-custom-vcl)). The appearance of the web interface controls may change from the defaults displayed once these services are enabled for your account.

About the All services page

The All services page displays a summary of all your services, sorted by requests per second. This page appears automatically when users with the appropriate access permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions) log in to the Fastly web interface. You can access the All services page by clicking the stopwatch icon.



You can click the Dashboard link to jump to a service's Dashboard page, or click a link in the CDN Configuration column to open the active configuration version page for a service. The Requests Per Second column shows the number of requests received per second for your service by Fastly, and the graphs in the Last 2 Hours of Requests column display the total number of requests received for your service by Fastly over a two hour time period.



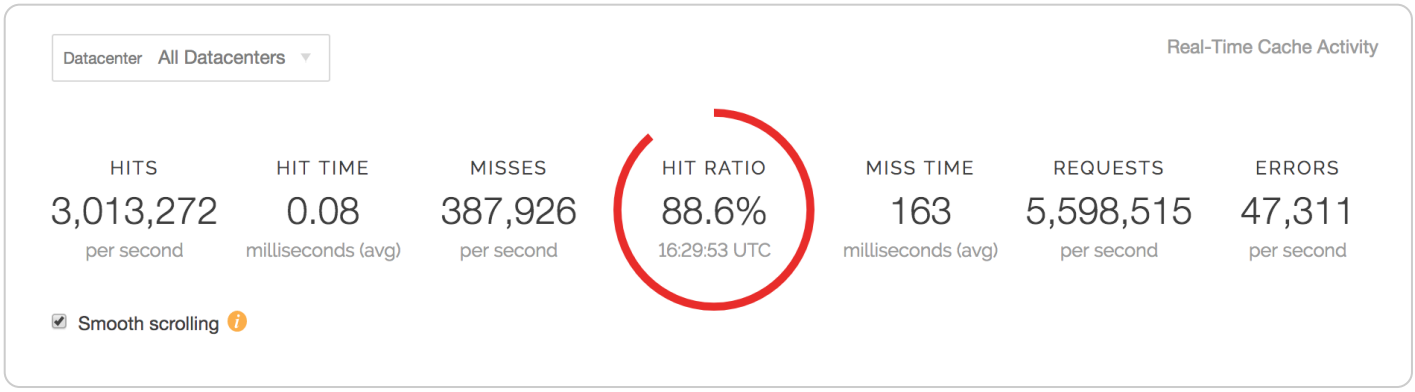
You can also search for a specific service associated with a domain by typing the domain name in the **Search by domain** field. The domain name you type must be an exact match to find the desired service.

About the Dashboard page

The Dashboard page allows you to separately monitor caching for each of your services in real time, as they operate on a second-by-second basis.



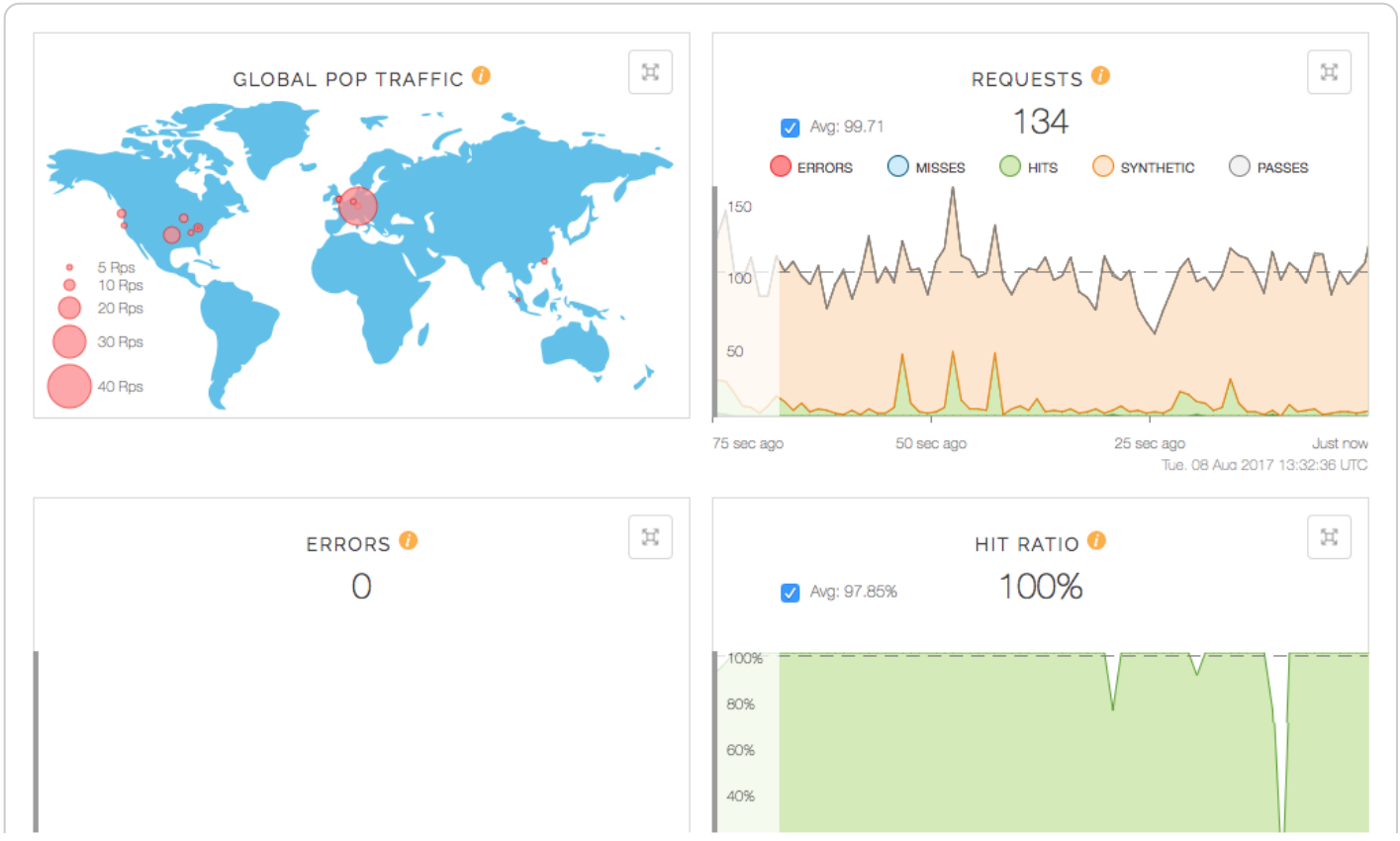
The Dashboard link appears automatically for logged in users with the appropriate access permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions); however, data on the page may appear grayed out or blank to some users, with no information displayed in the controls, when a customer's service does not receive enough requests for Fastly to display meaningful information about it in real time.

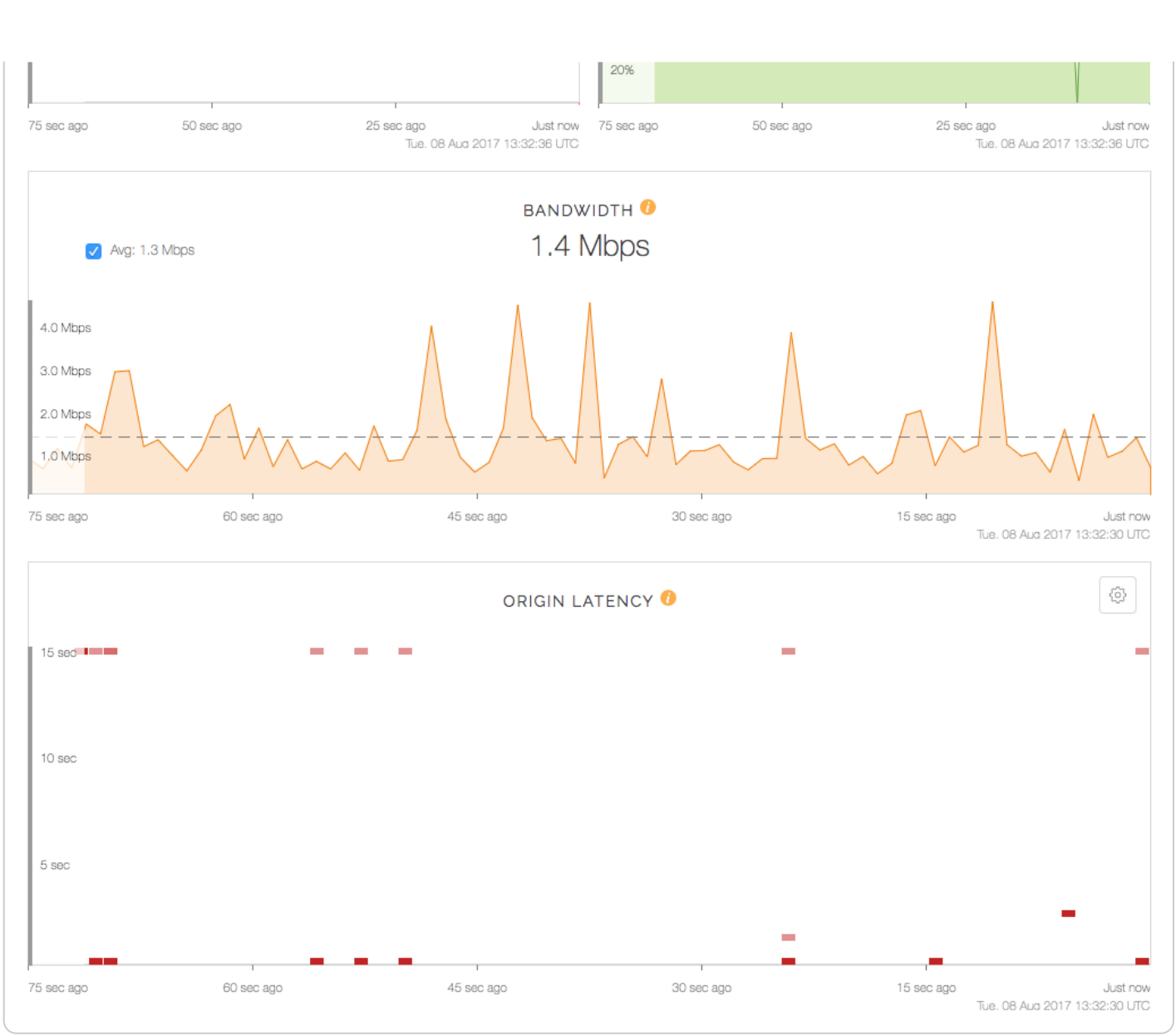


In addition to a menu allowing you to select the specific datacenter from which to view data (it defaults to data from all datacenters), the top of the dashboard includes the following real-time cache activity:

- **Hits:** the number of times requested data is found in cache
- **Hit Time:** the amount of time spent processing cache hits
- **Misses:** the number of times requested data is not found in cache
- **Hit Ratio:** the percentage of content being accessed that is currently cached by Fastly, defined as the proportion of cache hits to all cacheable content (hits + misses)
- **Miss Time:** the amount of time spent processing cache misses
- **Requests:** the total number of requests received for your site by Fastly
- **Errors:** the number of error requests that occurred

Below the real-time cache activity summary data, several graphs appear:





The graphed cache activity includes:

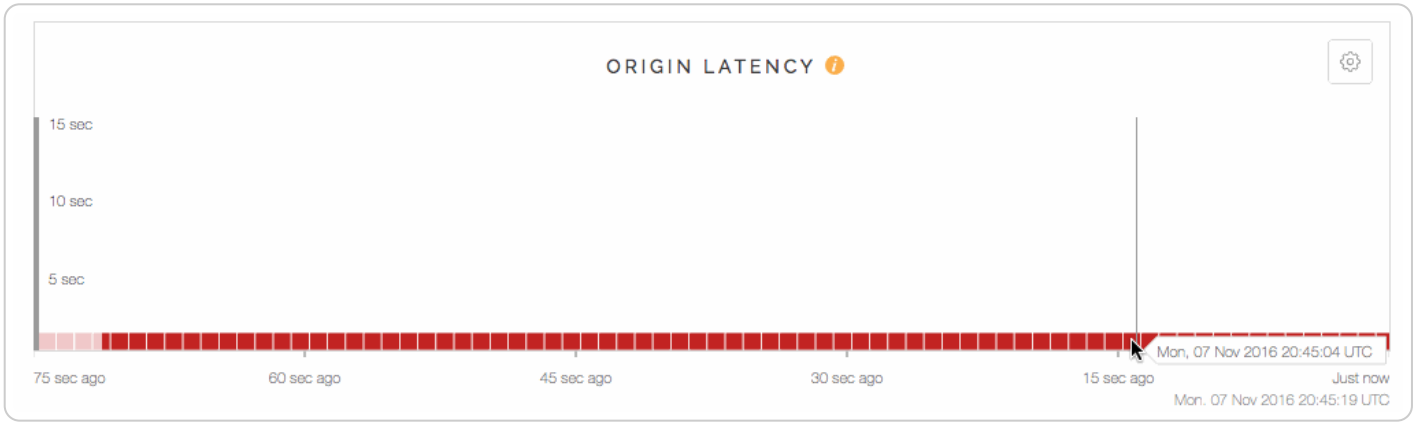
- **Global POP Traffic:** a heat map displaying global POP traffic through all POPs for your service.
- **Requests:** a graph displaying the total number of requests received for your site by Fastly over time.
- **Errors:** a graph displaying the number of error requests that occurred over time.
- **Hit Ratio:** a graph displaying the percentage of content being accessed that is currently cached by Fastly over time.
- **Bandwidth:** a graph displaying the bandwidth served from Fastly's servers to your website's visitors.
- **Origin Latency:** a histogram displaying the average amount of time to first byte (measured in milliseconds) on a cache miss. High origin latency means that your backends are taking longer to process requests.

One minute after Dashboard measurement data in these graphs rolls off the screen, it becomes available for retrieval on the Stats page (/guides/basic-concepts/about-the-web-interface-controls#about-the-stats-page). You may not see any traffic right away because of the following:

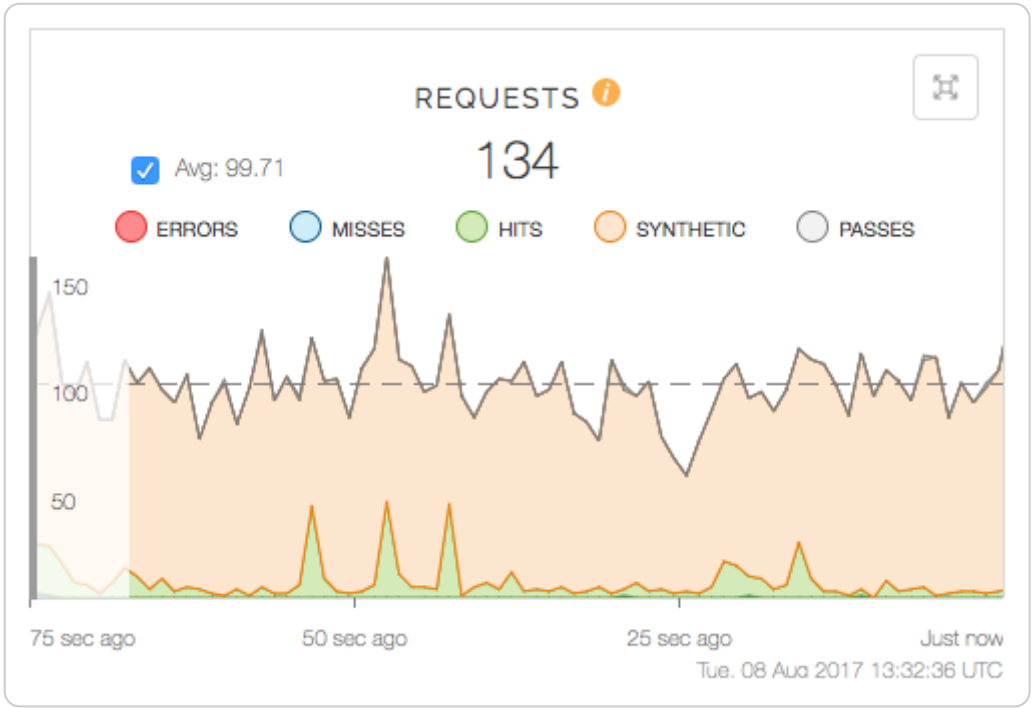
- **Not enough data is going to your site.** If this is the case, visit the site yourself to trigger some traffic.
- **You've made a CNAME change.** If this is the case, it could take from a few minutes to hours for the change to propagate your DNS servers. See how to edit your DNS record (/guides/basic-setup/adding-cname-records#updating-the-cname-record-with-your-dns-provider) to point to Fastly for more information.

Interacting with graphs

Hovering the cursor over any part of a graph displays a timestamp indicator that updates itself as you move the mouse.



The average line appears as a dashed line on some graphs. To hide the average line for a graph, deselect the **Avg** checkbox.



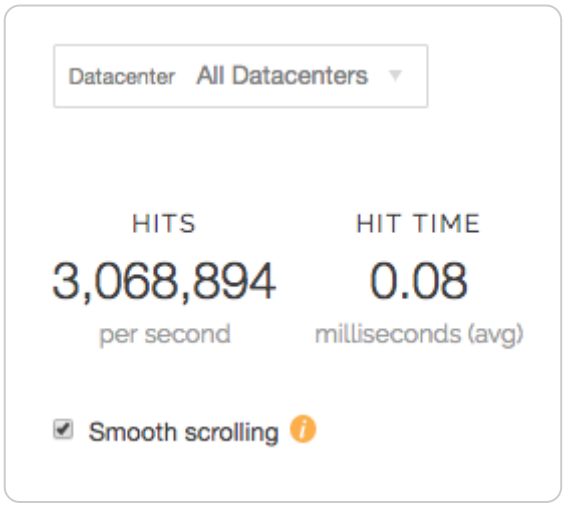
Resizing graphs

You can expand and minimize the view of some of the graphs using the quadruple arrow button in the right-hand corner of the graph to display an expanded view of the graph or special features it offers. Specifically:

- the Global POP Traffic heat map displays a larger view of the graph as well as the traffic in each POP region, with continuously updating data on the POP's current requests per second, the request error ratio, and the bandwidth going through that POP.
- the Requests, Errors, and Hit Ratio graphs expand to larger versions of themselves along with the already expanded versions of the Bandwidth and Origin Latency graphs.
- the Origin Latency graph specifically includes a small gear icon in the upper right corner that allows you to change the interval limit displayed by the graph from the default 15 second interval to a shorter time frame.

Disabling smooth scrolling

The Dashboard graphs update continuously. Leaving the Dashboard open for long periods of time, however, can occasionally lead to higher CPU utilization. To improve performance, you can deselect the **Smooth scrolling** checkbox. The graphing animations may not be as smooth when this checkbox is deselected.



About the Stats page

Fastly's Stats page provides a visual interface to our Stats API (/api/stats) that displays a series of graphs derived from your site's metrics statistical information. The Stats link appears automatically for logged in users with the appropriate access permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions).



The displayed caching and performance metrics help you optimize your website's speed. These metrics include the following:

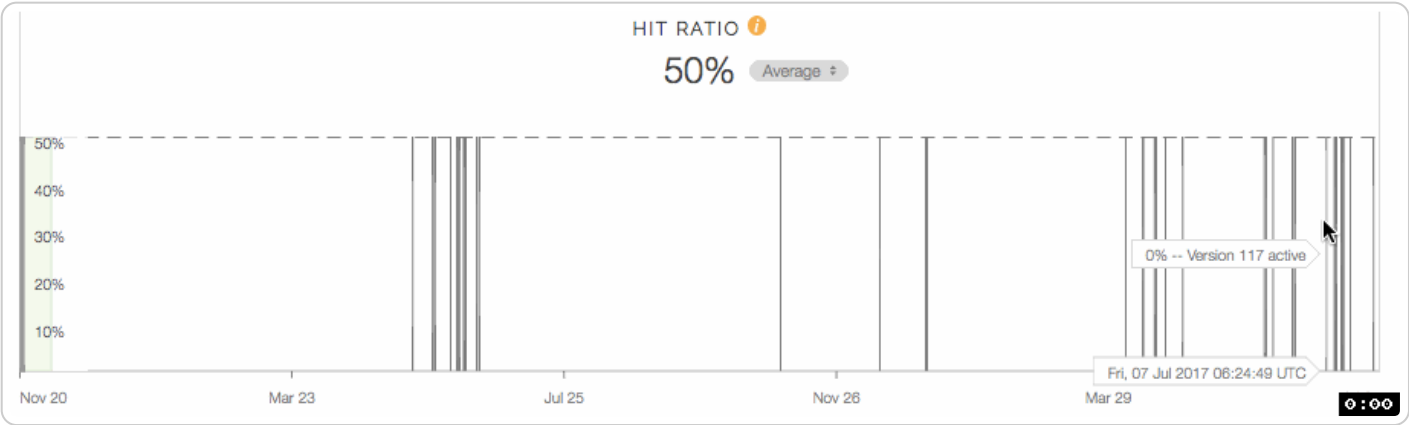
- **Hit Ratio** metrics tell you how well you are caching content using Fastly. This metric represents the proportion of cache hits versus all cacheable content (hits + misses). Increasing your hit ratio improves the overall performance benefit of using Fastly.
- **Cache Coverage** metrics show how much of your site you are caching with Fastly. This metric represents the ratio of cacheable requests (i.e., non "pass" requests) to total requests. Improving your cache coverage by reducing passes can improve site performance and reduce load on your origin servers.
- **Caching Overview** metrics compare Cache Hits, Cache Misses, Synthetic Responses (in VCL edge responses), and Passes (or requests that cannot be cached according to your configuration).

The traffic metrics analyze your website's traffic as it evolves over time. These metrics include the following:

- **Requests** metrics show you the total number of requests over time that were received for your site by Fastly.
- **Bytes Transferred** metrics show you the total number of bytes transferred by Fastly for your service.
- **Header & Body Bytes Transferred** metrics show you the relative values of bytes transferred when serving the body portion of HTTP requests and the header portion of the requests.
- **Miss Latency:** metrics show the distribution of the miss latency times for your origin.
- **Error Ratio** metrics show you the ratio of error responses (5XX status code errors (https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)) compared to the total number of requests for your site. This metric allows you to quickly identify error spikes at given times.
- **HTTP Info, Success, & Redirects** metrics shows the number of HTTP Info (1XX), Success (2XX), and Redirect (3XX) statuses served for your site using Fastly.
- **Status 3XX Details** metrics shows the breakdown between the number of HTTP Status 301s, 302s, 304s, and other 3XX requests.
- **HTTP Client and Server Errors** metrics shows the number of HTTP Client Errors (4XX), and Server Errors (5XX) served for your site by Fastly.

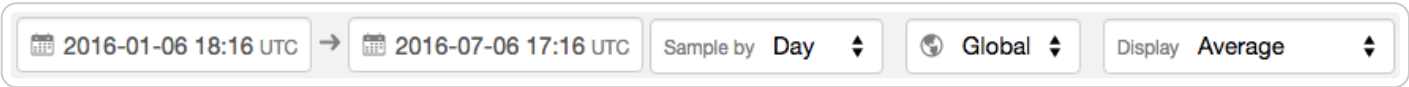
Interacting with Stats graphs

Service version activations appear as vertical lines on the Stats graphs. Hovering your cursor over any line displays the version's number and its activation timestamp.



Controlling the historical information displayed

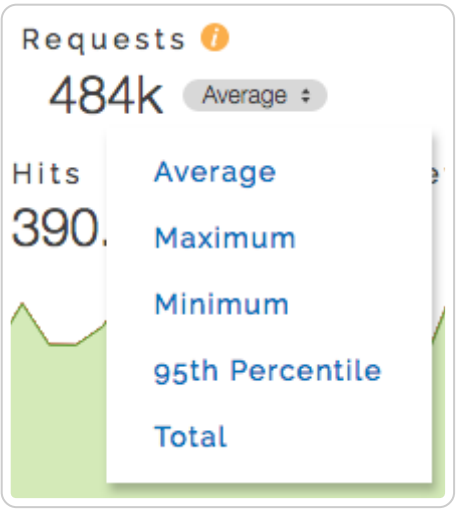
You can control how you view the historical information.



For all displayed graphs, you can choose:

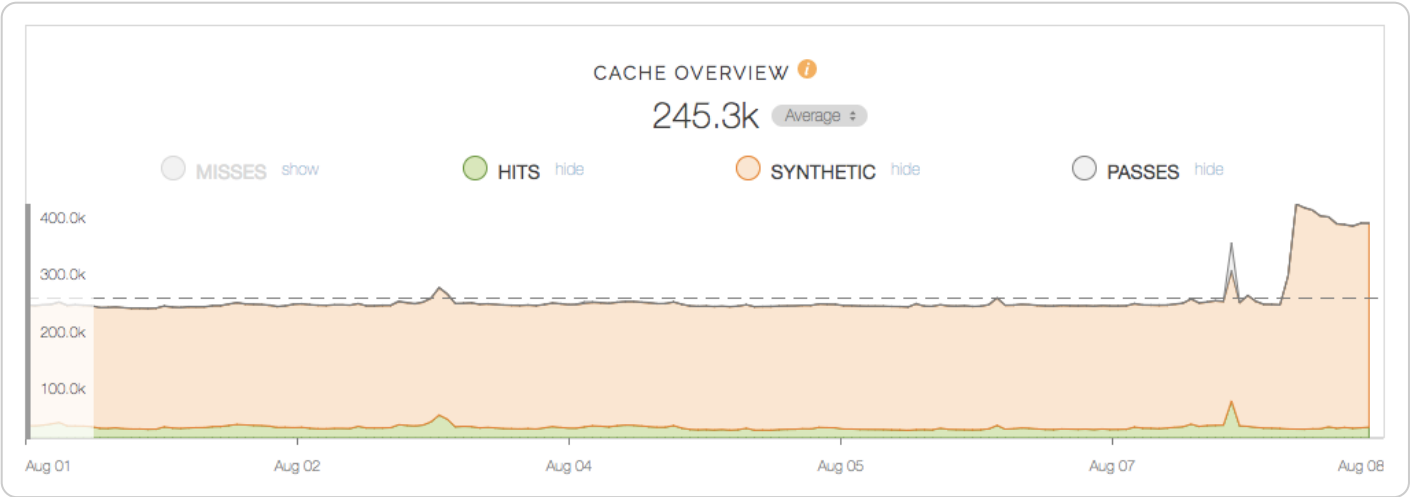
- the exact local date and time range of the graphed data
- how often to sample the data displayed
- whether to view global data for the graphs or only data from a specific region
- how to display the statistical values

You can change the statistics displayed in each graph. For example, notice the Average button on this Requests graph:



Clicking this button on any graph (including those on the Dashboard) lets you change the display of the graph's data to an average, a 95th percentile, a minimum, a maximum, or a total. When set to average, the graph displays the average as a dashed line.

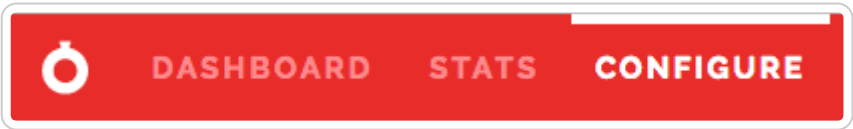
You can also exclude certain data entirely. For example, in this Caching Overview graph, hovering the cursor over the word "Hits" next to any of the data values displays a small, clickable **hide** link. Clicking this link will hide that value in the graph's display.



Notice that the actual numbers of the hidden data still appear grayed out in the controls, but the hide link changes to a show link and the graph itself doesn't display the hidden data at all.

About the Configure page

The Configure page allows you to define exactly how each instance of your cache should behave and deliver content from data sources. The Configure page appears automatically for logged in users with the appropriate access permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions).



You use the Configure page to create versions of each of your service's configuration settings and then use the controls to activate or deactivate (/guides/basic-setup/working-with-services) them.

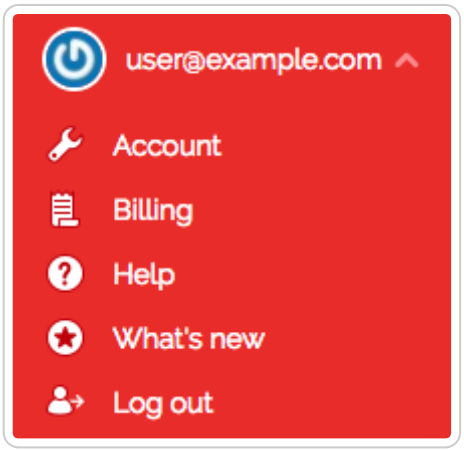
Specifically, you can configure and manage (/guides/basic-configuration/):

- the domains used to route requests to a service
- the hosts used as backends for a site and how they should be accessed
- the health checks that monitor backend hosts
- various request and cache settings, headers, and responses that control how Fastly caches and serves content for a service
- how logging (/guides/streaming-logs/setting-up-remote-log-streaming) should be performed and where server logs should be sent (as specified by an rsyslog endpoint)
- custom Varnish configuration language (/guides/vcl/) (VCL) files if custom VCL is enabled (/guides/vcl/uploading-custom-vcl)
- how conditions (/guides/conditions/) are mapped and used for a service at various times (e.g., during request processing, when Fastly receives a backend response, or just before an object is potentially cached)

With the appropriate permissions, you can activate configuration changes immediately and roll back those changes just as quickly should they not have the intended effect. The Configure page also allows you to compare differences (/guides/basic-setup/working-with-services#comparing-different-service-versions) between two configuration versions.

About the user menu

The user menu appears at the far right of the default control group:



Depending on your access permissions (/guides/user-access-and-control/configuring-user-roles-and-permissions), it provides access to a variety of account-specific and personal settings information. Specifically, it gives you access to Account details, Billing (/guides/account-types-and-billing/how-we-calculate-your-bill#reviewing-the-charges-to-your-account) information, Help information, and What's new references. It also provides a way for you log out of the web interface.

About the Account controls

Selecting Account from the user menu displays account-related details for your login including specifics about your Company Profile which include:

- **Company settings** where you'll find details about your company (e.g., its name and the phone number, the IP whitelist (/guides/account-management-and-security/enabling-an-ip-whitelist-for-account-logins)) as well as the location to cancel your account (/guides/account-types-and-billing/accounts-and-pricing-plans#canceling-your-account)
- **Customer options** like the setting for company-wide two-factor authentication (/guides/account-management-and-security/enabling-and-disabling-two-factor-authentication#managing-two-factor-authentication-as-a-company)
- **User management** controls where you can control user invitations and roles (/guides/user-access-and-control/)
- **Account API tokens** created by users within your account to control or restrict access (/guides/account-management-and-security/using-api-tokens) to various services

- **Billing** controls where you can review the charges to your account (/guides/account-types-and-billing/how-we-calculate-your-bill#reviewing-the-charges-to-your-account), change your credit card information (/guides/account-types-and-billing/paying-your-bill#changing-your-credit-card-information), and update your company's tax address (/guides/account-types-and-billing/paying-your-bill#changing-your-tax-or-billing-address)

You'll also find Personal Profile information here. Specifically:

- **Your profile** including your name and your email (/guides/user-access-and-control/email-and-password-changes#changing-your-name-or-email-address) address
- **Change password** controls that allow you to update your current login password (/guides/user-access-and-control/email-and-password-changes#changing-your-password)
- **Two-factor authentication** information where you can manage the multi-factor authentication (/guides/account-management-and-security/enabling-and-disabling-two-factor-authentication#managing-two-factor-authentication-as-a-user) controls for your personal login
- **Personal API tokens** where you can create and delete your personal API tokens (/guides/account-management-and-security/using-api-tokens) you need to control access to various services and resources within your Fastly account

About the Help page

The Help page allows you to search Fastly documentation for immediate help with questions you may have about using and accessing Fastly services. If you can't find what you need in the documentation, you can also use this page to submit a request (/guides/detailed-product-descriptions/about-fastlys-customer-support) for help directly from Fastly Customer Support.

About the What's new panel

The What's new panel provides a brief overview of the changes most recently made to the Fastly web interface.

---

§ Content and its delivery (/guides/basic-concepts/content-and-its-delivery)

Content types delivered by Fastly

The underlying protocol used by the World Wide Web to define how content is formatted and transmitted is called the Hypertext Transfer Protocol (HTTP). Fastly's CDN Service delivers all HTTP-based file content (e.g., HTML, GIF, JPEG, PNG, JavaScript, CSS) including the following:

- Static content
- Dynamic content
- Video content

Each content type is described below.

Static content

Static content includes content that remains relatively unchanged. Fastly can control static content in two ways:

- using the time to live (TTL) method, where Fastly's cache re-validates the content after expiration of the TTL, or
- using Fastly's Instant Purge functionality, in which content remains valid until the cache receives a purge request (/guides/purging/) that invalidates the content.

Examples of static content include images, css, and javascript files.

Dynamic content

Dynamic content includes content that changes at unpredictable intervals, but can still be cached for a fraction of time. We serve this dynamic content by taking advantage of Fastly's Instant Purge functionality. Using this functionality, dynamic content remains valid only until a Fastly cache receives a purge request (/guides/purging/) that invalidates the content. Fastly understands that the rate of those purge requests cannot be predicted. Dynamic content may change frequently as a source application issues purge requests in rapid succession to keep the content up to date. Dynamic content can, however, remain valid for months if there are no changes requested.

Examples of dynamic content include sports scores, weather forecasts, breaking news, user-generated content, and current store item inventory.

Video content

Video content includes:

- Live video streams
- Video on Demand (VOD) content libraries

Video content can be served using standard HTTP requests. Specifically, Fastly supports HTTP Streaming standards, including HTTP Live Streaming (HLS), HTTP Dynamic Streaming (HDS), HTTP Smooth Streaming (HSS), and MPEG-DASH. For Fastly's CDN Service to deliver video, the video must be packaged.

Content sources supported by Fastly

Fastly caches deliver various types of content from many different sources. Supported sources include:

- Websites
- Internet APIs
- Internet Applications
- Live and Live Linear Video
- Video on Demand (VOD) Libraries

Regardless of the content source, the content's source server must communicate using HTTP. HTTP defines specific types of "methods" that indicate the desired action to be performed on content. The manner in which those HTTP methods are used (the standard, primary methods being GET, POST, PUT, and DELETE) can be labeled as being RESTful (https://en.wikipedia.org/wiki/Representational_state_transfer) or not. Fastly supports RESTful HTTP by default, but also can support the use of non-RESTful HTTP as long as the method used is mapped to its appropriate cache function. Each of the content sources supported by Fastly are described in more detail below.

Websites

Websites are servers that provide content to browser applications (e.g., Google's Chrome, Apple's Safari, Microsoft's Internet Explorer, Opera Software's Opera) when end users request that content. The content contains both the requested data and the formatting or display information the browser needs to present the data visually to the end user.

With no CDN services involved, browsers request data by sending HTTP GET requests that identify the data with a uniform resource locator (URL) address to the origin server that has access to the requested data. The server retrieves the data, then constructs and sends an HTTP response to the requestor. When a CDN Service is used, however, the HTTP requests go to the CDN rather than the origin server because the customer configures it to redirect all requests for data to the CDN instead. Customers do this by adding a CNAME or alias for their origin server that points to Fastly instead.

Internet APIs

Application program interfaces (APIs) serve as a language and message format that defines exactly how a program will interact with the rest of the world. APIs reside on HTTP servers. Unlike the responses from a website, content from APIs contain only requested data and identification information for that data; no formatting or display information is included. Typically the content serves as input to another computing process. If it must be displayed visually to an end user, a device application (such as, an iPad, Android device, or iPhone Weather application) does data display instead.

Legacy internet applications

Legacy Internet applications refer to applications not originally developed for access over the Internet. These applications may use HTTP in a non-RESTful manner. They can be incrementally accelerated without caching, benefiting only from the TCP Stack optimization done between edge Fastly POPs and the Shield POP, and the Shield POP to the origin. Then caching can be enabled incrementally, starting with the exchanges with the greatest user-experienced delay.

Live and live linear video streams & video on demand libraries

Live and live linear video content (for example, broadcast television) is generally delivered as a "stream" of information to users, which they either choose to watch or not during a specific broadcast time. Video on demand (VOD), on the other hand, allows end users to select and watch video content when they choose to, rather than having to watch at a specific broadcast time.

Regardless of which type of video content an end user experiences, a video player can begin playing before its entire contents have been completely transmitted. End users access the video content from a customer's servers via HTTP requests from a video player application that can be embedded as a part of a web browser. Unlike other types of website content, this content does not contain formatting or display information. The video player handles the formatting and display instead.

When the video content is requested, the customer's server sends the content as a series of pre-packaged file chunks along with a manifest file required by the player to properly present the video to the end user. The manifest lists the names of each file chunk. The video player application needs to receive the manifest file first in order to know the names of the video content chunks to request.

"Pre-packaging" in this context refers to the process of receiving the video contents, converting or "transcoding" the stream into segments (chunks) for presentation at a specific dimension and transmission rate, and then packaging it so a video player can identify and request the segments of the live video a user wants to view.

To request video delivery on your account, contact your Fastly Account Representative at sales@fastly.com (mailto:sales@fastly.com).

---

§ Fastly POP locations (/guides/basic-concepts/fastly-pop-locations)

Our points of presence (POPs) on the Internet are strategically placed at the center of the highest density Internet Exchange Points around the world. Fastly's Network Map (https://www.fastly.com/network-map) shows a detailed view of the current and planned locations

of all Fastly POPs. In addition, our datacenter API endpoint (/api/tools#datacenter) provides a list of all Fastly POPs, including their precise latitude and longitude locations.

Once you're signed up for Fastly service (either through a test account (/guides/account-types-and-billing/accounts-and-pricing-plans) or a paid plan) you can a see a live, real-time visual representation (/guides/basic-concepts/about-the-web-interface-controls#about-the-dashboard-page) of the general regions of the world in which Fastly's points of presence (POPs) receive requests for your service.

Will Fastly ever adjust POP locations or service regions? How will I be notified?

Fastly continues to grow its network footprint, adding new service POPs in the process. At times, expansion may result in the addition of new billable regions (/guides/account-types-and-billing/how-we-calculate-your-bill#reviewing-the-charges-to-your-account) to our network. We'll announce new POP locations and new billable regions in advance through our network status page (/guides/debugging/fastlys-network-status) at status.fastly.com (https://status.fastly.com/). Contact sales@fastly.com (mailto:sales@fastly.com) with specific contract or billing questions.

---

§ How caching and CDNs work (/guides/basic-concepts/how-caching-and-cdns-work)

Fastly is a Content Delivery Network, or CDN. CDNs work on the principle that once a piece of content has been generated it doesn't need to be generated again for a while so a copy can be kept around in a cache. Cache machines are optimized to serve small files *very very* quickly. CDNs typically have caches placed in datacenters all around the world. When a user requests information from a customer's site they're actually redirected to the set of cache machines closest to them instead of the customer's actual servers. This means that a European user going to an American site gets their content anywhere from 200-500ms faster. CDNs also minimize the effects of a cache miss. A cache miss occurs when a user requests a bit of content and it is not in the cache at that moment (because it's expired, because no-one has asked for it before, or because the cache got too full and old content was thrown out).

What can be cached?

CDNs are good at managing a cache of small, static resources (for example, static images, CSS files, Javascripts, and animated GIFs). CDNs are also popular for offloading expensive-to-serve files like video and audio media.

At Fastly, our architecture (known as a *reverse proxy*) is designed to enable customers to go a step further and cache entire web pages for even more efficient handling of your traffic.

> ★ **TIP:** Static files + media objects + web pages = your whole site. With the right service configuration (which we can assist you in setting up) Fastly can reduce your backend traffic by orders of magnitude with no loss in control over the content your users see.

Managing the Cache

Caching serves as a powerful weapon in your make-the-site-faster arsenal. However, most objects in your cache aren't going to stay there permanently. They'll need to expire so that fresh content can be served. How long that content should stay in the cache might be mere seconds or a number of minutes or even a year or more.

How can you manage which of your content is cached, where, and for how long? By setting policies that control the cached data. Most caching policies are implemented as a set of HTTP headers sent with your content by the web server (as specified in the configuration or the application). These headers were designed with the client (browser) in mind but CDNs like Fastly will also use those headers as a guide on caching policy.

Expires

The `Expires` header is the original cache-related HTTP header and tells the cache (typically a browser cache) how long to hang onto a piece of content. Thereafter, the browser will re-request the content from its source. The downside is that it's a static date and if you don't update it later, the date will pass and the browser will start requesting that resource from the source every time it sees it.

If none of the following headers are found in the request, Fastly will respect the `Expires` header value.

Cache-Control

The `Cache-Control` headers (introduced in the HTTP 1.1 specification) cover browser caches and in most cases, intermediate caches as well:

- `Cache-Control: public` - Any cache can store a copy of the content.
- `Cache-Control: private` - Don't store, this is for a single user.
- `Cache-Control: no-cache` - Re-validate before serving this content.
- `Cache-Control: no-store` - Don't ever store this content.
- `Cache-Control: public, max-age=[seconds]` - Caches can store this content for *n* seconds.

- `Cache-Control: s-maxage=[seconds]` - Same as max-age but applies specifically to proxy caches.

Only the `max-age`, `s-maxage`, and `private` Cache-Control headers will influence Fastly's caching. All other Cache-Control headers will not, but will be passed through to the browser. For more in-depth information about how Fastly responds to these Cache-Control headers and how these headers interact with Expires and Surrogate-Control, check out our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

> ℹ **NOTE:** For more information on the rest of the Cache-Control headers, see the relevant section in Mark Nottingham's Caching Tutorial (https://www.mnot.net/cache_docs/#CACHE-CONTROL).

Surrogate Headers

`Surrogate` headers are a relatively new addition to the cache management vocabulary (described in this W3C tech note (http://www.w3.org/TR/edge-arch/)). These headers provide a specific cache policy for proxy caches in the processing path. `Surrogate-Control` accepts many of the same values as `Cache-Control`, plus some other more esoteric ones (read the tech note for all the options).

One use of this technique is to provide conservative cache interactions to the browser (for example, `Cache-Control: no-cache`). This causes the browser to re-validate with the source on every request for the content. This makes sure that the user is getting the freshest possible content. Simultaneously, a `Surrogate-Control` header can be sent with a longer `max-age` that lets a proxy cache in front of the source handle most of the browser traffic, only passing requests to the source when the proxy's cache expires.

With Fastly, one of the most useful `Surrogate` headers is `Surrogate-Key`. When Fastly processes a request and sees a `Surrogate-Key` header, it uses the space-separated value as a list of tags to associate with the request URL in the cache. Combined with Fastly's Purge API (/api/purge) an entire collection of URLs can be expired from the cache in one API call (and typically happens in around 1ms). `Surrogate-Control` is the most specific.

Fastly and Cache Control Headers

Fastly looks for caching information in each of these headers as described in our Cache-Control docs (/guides/tutorials/cache-control-tutorial). In order of preference:

- `Surrogate-Control:`
- `Cache-Control: s-maxage`

- `Cache-Control: max-age`
- `Expires:`

Shielding

When an object or collection of objects in the cache expires, the next time any of those objects are requested, the request is going to get passed through to your application. Generally, with a good caching strategy, this won't break things. However, when a popular object or collection of objects expires from the cache, your backend can be hit with a large influx of traffic as the cache nodes refetch the objects from the source.

In most cases, the object being fetched is not going to differ between requests, so why should every cache node have to get its own copy from the backend? With Shield Nodes, they don't have to. Shielding configured through the Fastly web interface (/guides/performance-tuning/shielding) allows you to select a specific datacenter (most efficiently, one geographically close to your application) to act as a shield node. When objects in the cache expire, the shield node is the only node to get the content from your source application. All other cache nodes will fetch from the shield node, reducing source traffic dramatically.

Resources

- Wikipedia: Reverse Proxy (https://en.wikipedia.org/wiki/Reverse_proxy)
- Fastly's Cache-Control docs (/guides/tutorials/cache-control-tutorial)
- Mark Nottingham's Caching Tutorial (https://www.mnot.net/cache_docs/#CACHE-CONTROL)
- Surrogate header W3C tech note (http://www.w3.org/TR/edge-arch/)

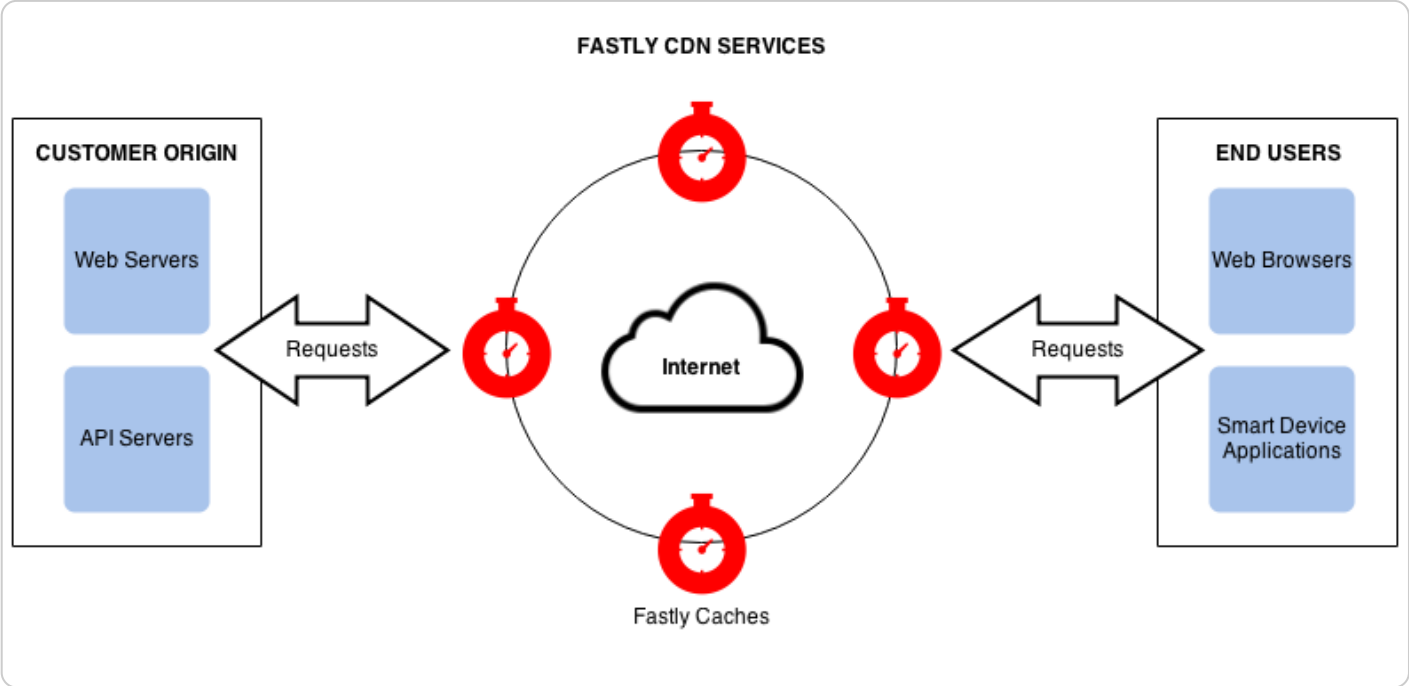
---

§ How Fastly's CDN Service works (/guides/basic-concepts/how-fastlys-cdn-service-works)

Fastly is a content delivery network (https://en.wikipedia.org/wiki/Content_delivery_network) (CDN). We serve as an Internet intermediary and offer the Fastly CDN Service to make transmission of your content to your end users more efficient.

You can make content available through your websites and Internet-accessible (hosted) application programming interfaces (APIs). You can create content (customer-generated content), as can your end users (user-generated content). Fastly's CDN Service then makes the

transmission of that content (which we sometimes refer to as "content objects") more efficient by automatically storing copies at intermediate locations on a temporary basis. The process of storing these copies is known as "caching" and the server locations in which they are stored are referred to as "caches."



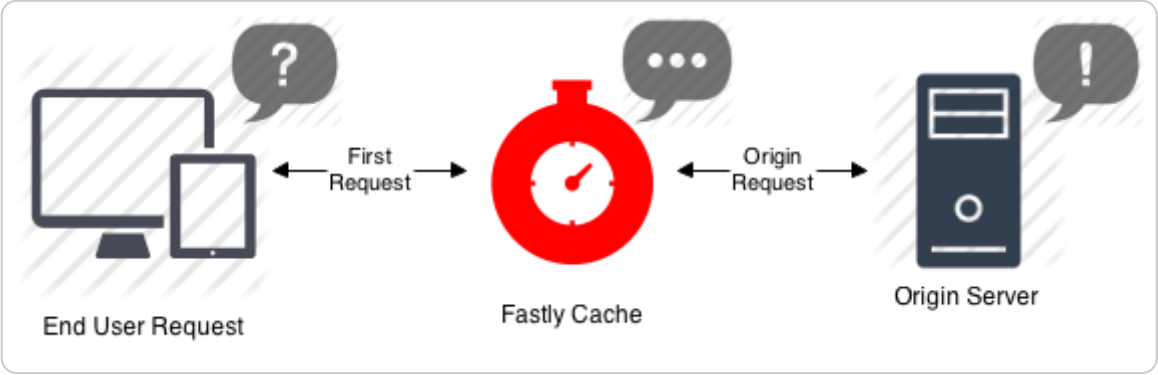
Fastly's delivers its CDN service from key access points to the Internet called "points of presence" (POPs). Fastly places POPs (/guides/basic-concepts/fastly-pop-locations) where their connectivity to the Internet reduces network transit time when delivering content to end-users. Each POP has a cluster of Fastly cache servers. When end users request your content objects, Fastly delivers them from whichever of the cache locations are closest to each end user.

Fastly's caches only receive and process your end user requests for content objects. You decide which objects will be cached, for how long, who can access them, whether they are to be encrypted when transmitted over the Internet, and when the objects will be deleted from the caching service. You make these decisions by specifically configuring Fastly's CDN Service with these requirements. We refer to this configuration process as "provisioning."

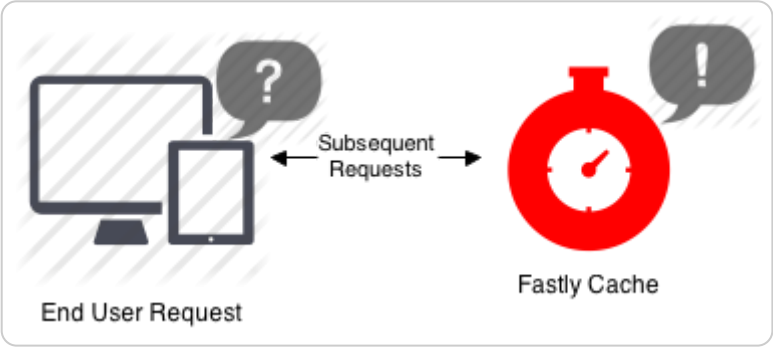
To provision Fastly's CDN service (/guides/basic-setup/sign-up-and-create-your-first-service), you must identify which of your application servers will provide the original content objects for each of your various domains (e.g., company.com, myco.com). Your application servers can be physical servers in a datacenter or hosting facility, or applications running on cloud services like Amazon, or any combination. Fastly refers to these source servers as "origin" and "backend" servers interchangeably.

The first time each Fastly cache receives a request for a content object, it fetches the object from the appropriate origin server. If multiple origin servers are specified, the cache will distribute the processing load for the fetches across all of them (based on the configuration criteria set by you).

After the content object is fetched, the cache stores a copy of it and forwards its response to the end user.



Each time after the first time an end user requests that same content object, the Fastly cache fulfills requests by retrieving the cached copy from storage (or memory) and immediately delivering it to the end user – the fetch step to the original copy is not repeated until the content object either expires or becomes invalidated.



Can Fastly host my content?

We accelerate your site by caching both static assets and dynamic content by acting as a reverse proxy (https://en.wikipedia.org/wiki/Reverse_proxy) to your origin server (also known as "Origin Pull"), but we do not provide services for uploading your content to our servers.

In addition to using your own servers as the source, we also support various "cloud storage" services as your origin, such as Amazon Simple Storage Service (/guides/integrations/amazon-s3) (S3), Google Cloud Storage (/guides/integrations/google-cloud-storage) (GCS), and Google Compute Engine (/guides/integrations/google-compute-engine) (GCE) as your file origin. Our partnership with Google (https://www.fastly.com/partner/gcp) in particular enables us to have direct connectivity to their cloud infrastructure.

---

§ HTTP status codes cached by default (/guides/basic-concepts/http-status-codes-cached-by-default)

Fastly caches the following response status codes by default. In addition to these statuses, you can force an object to cache under other states using conditions (/guides/conditions/) and responses (/guides/basic-configuration/responses-tutorial).

Code	Message
`200`	OK
`203`	Non-Authoritative Information
`300`	Multiple Choices
`301`	Moved Permanently
`302`	Moved Temporarily
`404`	Not Found
`410`	Gone

§ Self-provisioned Fastly services (/guides/basic-concepts/self-provisioned-fastly-services)

You can configure or "provision" Fastly caching and video services personally, independent of Fastly staff, via the Fastly web interface (https://manage.fastly.com). Fastly calls this "self-provisioning." Self-provisioning tasks include things like:

- creating and activating services
- adding domains and origin servers
- configuring load balancing
- modifying how services handle HTTP headers
- submitting purge commands

Once provisioned, Fastly services can be activated immediately. If self-provisioned tasks fail to operate in an appropriate or expected manner, you can find answers to a variety of frequently asked questions in Fastly's guides and tutorials (/guides/). You can also receive personalized assistance by submitting requests (/guides/detailed-product-descriptions/about-fastlys-customer-support) directly to Fastly's Customer Support staff.

§ Enabling API caching (/guides/api-caching/enabling-api-caching)

Application Programming Interfaces (APIs) allow you to retrieve data from a variety of web services. Fastly makes it possible for you to cache your API so you can accelerate the performance of your service-oriented architecture. It optimizes your API's performance by efficiently handling traffic bursts and reducing latency.

An example

Let's look at an example to learn how API caching works. Imagine we're an online magazine with articles on which users can make comments. Each article can have many comments, and each comment is authored by exactly one user.



We'll design a RESTful API specification and use it to manipulate and retrieve comments:

- `GET /comment` - Returns a list of all comments
- `GET /comment/:id` - Returns a comment with the given ID
- `POST /comment` - Creates a new comment
- `PUT /comment/:id` - Updates a comment with the given ID
- `DELETE /comment/:id` - Deletes a comment with the given ID

The create, read, update, and delete (CRUD) methods ensure the API can perform its basic operations, but they don't expose the relational aspect of the data. To do so, you would add a couple of relational endpoints:

- `GET /articles/:article_id/comments` - Get a list of comments for a given article
- `GET /user/:user_id/comments` - Get all comments for a given user

Endpoints like these allow programmers to get the information they need to do things like render the HTML page for an article, or display comments on a user's profile page. While there are many other possible endpoints we could construct, this set should suffice for the purposes of this guide. Let's assume that the API has been programmed to use an Object-Relational Mapper (ORM), such as ActiveRecord, when interacting with the database.

Determining which API endpoints to cache

Start by identifying the URLs you want to cache. We recommend splitting the specification endpoints into two groups.

The first group, called "accessors," retrieves or accesses the comment data. These are the endpoints you want to cache using Fastly. Using the example, four endpoints match this description:

- `GET /comment`
- `GET /comment/:id`
- `GET /article/:article_id/comments`
- `GET /user/:user_id/comments`

The second group, called "mutators," changes or mutates the comment data. These endpoints are always dynamic, and are therefore uncacheable. Using the example, three endpoints match this description:

- `POST /comment`
- `PUT /comment/:id`
- `DELETE /comment/:id`

You should see a pattern emerging. Because the example API is RESTful, we can use a simple rule to identify the accessor and mutator endpoints: GET endpoints can be cached, but PUT, POST, and DELETE endpoints cannot.

Once you've gathered this information, you're ready to program the API to configure PURGE requests.

Configuring PURGE requests

Don't be tempted to point at the PUT, POST, and DELETE endpoints as the place where data is modified. In most modern APIs, these endpoints represent an interface to the actual model code responsible for handling the database modifications.

In the example, we assumed that we'd be using an ORM to perform the actual database work. Most ORMs allow programmers to set special "callbacks" on models that will fire when certain actions have been performed (e.g., before or after validation, or after creating a new record).

For purging, we are interested in whether a model has saved information to the database — whether it's a new record, an update to an existing record, or the deleting of a record. At this point, we'd add a callback that tells the API to send a PURGE request to Fastly for each of the cacheable endpoints.

For an ActiveRecord comments model, you could do something like this:

```
require 'fastly'

class Comment < ActiveRecord::Base
  fastly = Fastly.new(api_key: 'FASTLY_API_TOKEN')

  after_save do
    fastly.purge "/comment"
    fastly.purge "/comment/#{self.id}"
    fastly.purge "/article/#{self.article_id}/comments"
    fastly.purge "/user/#{self.user_id}/comments"
  end
end
```

Keep two things in mind when creating the callback:

- The purge code should be triggered *after* the information has been saved to the database, otherwise a race condition could be created where Fastly fetches the data from the origin server before the data has been saved to the database. This would cache the old data instead of the new data.
- These URLs are being purged because they have content that changes when a comment is changed.

With the model code in place, the API is now ready to be cached.

Setting up Fastly

The final step to enabling API caching involves setting up Fastly. You'll need to:

- Create a new service
- Add the domain for the API
- Add the origin server that powers the API

In addition, you can optionally create rules that tell Fastly how to work with the specific elements that are exclusive to your API.

> ℹ **NOTE:** By default, Fastly will not cache PUT, POST, and DELETE requests. For more information, see our guide on default caching behavior of HTTP methods (/guides/debugging/using-get-instead-of-head-for-command-line-caching-tests).

Creating a new service

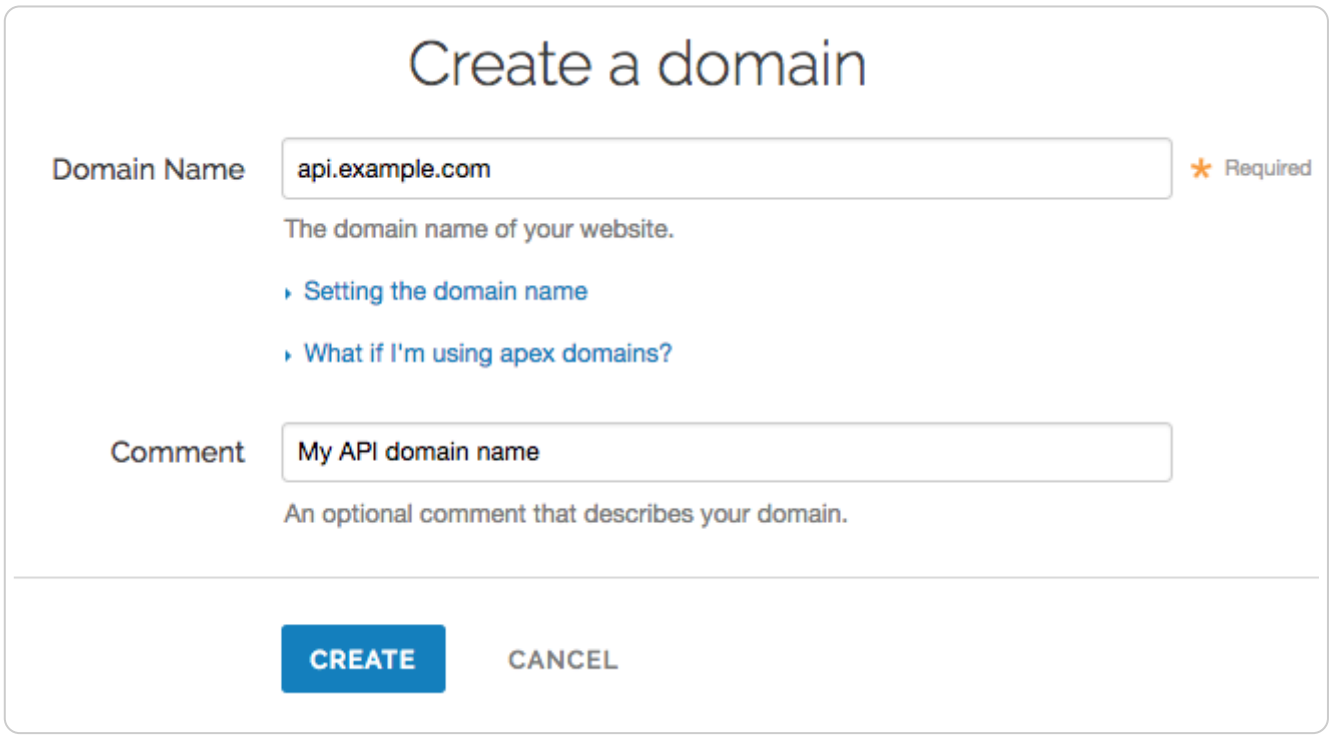
Follow the instructions for creating a new service (/guides/basic-setup/working-with-services#creating-a-new-service). You'll add specific details about your API server when you fill out the **Create a new service** fields:

- In the **Name** field, type a name for this service that helps you identify it's related to caching your API information (e.g., `My API Service`).
- In the **Domain** field, type the domain name associated with your API (e.g., `api.example.com`).
- In the **Address** field, type the IP address or hostname of your API server.

Adding the domain

Follow these instructions to add the API's domain name to your Fastly service:

1. On the **Configure** page, click the **Configuration** button and then select **Clone active**. The Domains page appears.
2. Click the **Create domain** button. The Create a domain page appears.



3. Fill out the **Create a domain** fields as follows:
   - In the **Domain Name** field, type the domain name for the API.
   - In the **Comment** field, type an optional comment that describes your domain.
4. Click **Create**. Your API's domain name appears in the list of domains.

Adding the origin server

Follow the instructions for connecting to origins (/guides/basic-configuration/connecting-to-origins). You'll add specific details about your API server when you fill out the **Create a host** fields:

- In the **Name** field, type a name for the origin server that helps you identify it's related to caching your API information.
- In the **Address** field, type the IP address (or hostname) of the API server.

---

§ Implementing API cache control (/guides/api-caching/implementing-api-cache-control)

This guide explains how to implement API cache control. Once you've enabled API caching (/guides/api-caching/enabling-api-caching), and ensured purging works properly with your cached data, you can set up specific headers like cache-control and surrogate-control to change when data is cached.

Understanding cache control headers

In general, we assume that GET requests are cached and PUT, POST, and DELETE requests are not. For an ideal REST API, this rule works well. Unfortunately, most APIs are far from ideal and require additional caching rules for some requests.

For these reasons, it's a good idea to set cache-control headers when migrating APIs to Fastly. Cache-control, as defined by RFC 7234 (https://tools.ietf.org/html/rfc7234#section-5.2) (the HTTP specification), includes many different options for appropriate handling of cached data. Specifically, cache-control headers tell user agents (e.g., web browsers) how to handle the caching of server responses. For example:

- `Cache-Control: private`
- `Cache-Control: max-age=86400`

In the first example, `private` tells the user agent the information is specific to a single user and should not be cached for other users. In the second example, `max-age=86400` tells the user agent the response can be cached, but that it expires in exactly 86,400 seconds (one day).

Fastly respects cache-control headers by default, but you can also use another proxy-specific header: surrogate-control. Surrogate-control headers are similar to cache-control headers, but provide instructions to reverse proxy caches like Fastly. You can use cache-control and surrogate-control headers together. For more information about cache-control and surrogate-control headers, see our cache control tutorial (/guides/tutorials/cache-control-tutorial).

An updated example

Let's take a look at how the cache-control headers could be used in our original example, the comments API (/guides/api-caching/enabling-api-caching#an-example). Recall the API endpoint that provided a list of comments for a given article:

`GET /article/:article_id/comments`

When a user submits a comment for a given article, the response from this endpoint will be purged from the Fastly cache by the comment model (/guides/api-caching/enabling-api-caching#configuring-purge-requests). It's hard to predict when content will change. Therefore, we'd like to ensure the following:

1. If the content doesn't change, it should stay cached in Fastly for a reasonable amount of time.
2. If the content does change, it should not be cached by the client longer than it needs to be.

The goal is to ensure that API responses will reach clients in a timely manner, but we also want to ensure that clients always have the most up-to-date information. The first constraint can be solved by using the surrogate-control header, and the second constraint can be solved by using the cache-control header:

```
Surrogate-Control: max-age=86400
Cache-Control: max-age=60
```

These headers tell Fastly that it is allowed to cache the content for up to one day. In addition, the headers tell the client that it is allowed to cache the content for 60 seconds, and that it should go back to its source of truth (in this case, the Fastly cache) after 60 seconds.

Implementing cache control

Migrating APIs isn't easy, even for experienced teams. When migrating an API to Fastly, we recommend separating the task into three strategic endpoint migrations to make the process more manageable while still maintaining the validity of the API as a whole.

Preparing the API

To ensure that the API bypasses the cache during the piecewise migration, we must have every API endpoint return a specific control header:

`Cache-Control: private`

This header tells Fastly that a request to any endpoint on the API should bypass the cache and be sent directly to the origin. This will allow us to serve the API via Fastly and have it work as expected.

NOTE: Modern web frameworks allow for blanket rules to be overridden by specific endpoints (for example, by the use of middlewares). Depending on how the API has been implemented, this step might be as simple as adding a single line of code.

Serving traffic with Fastly

The next step is configuring a Fastly service (/guides/api-caching/enabling-api-caching#setting-up-fastly) to serve the API's traffic. After you save the configuration, there will be an immediate speed improvement. This happens because Fastly's cache servers keep long-lasting connections to the API's origin servers, which reduces the latency overhead of establishing multiple TCP connections.

Migrating endpoints

Now we can implement instant purge caching for each cacheable API endpoint, one at a time. The order in which this is done depends on the API, but by targeting the slowest endpoints first, you can achieve dramatic improvements for endpoints that need them the most. Because each endpoint can be worked on independently, the engineering process is easier to manage.

Excluding endpoints

The last step is deciding which API endpoints you don't want Fastly to cache. To disable caching for endpoints, you'll need to add new conditions for the endpoints (/guides/performance-tuning/controlling-caching#conditionally-preventing-pages-from-caching). As you learned in Preparing the API, using the `Cache-Control: private` header is another option for disabling caching.

---

§ Purging API cache with surrogate keys (/guides/api-caching/purging-api-cache-with-surrogate-keys)

Fastly makes it possible for you to cache your API so you can accelerate the performance of your service-oriented architecture. Of course, caching your API is one thing - efficiently invalidating the API cache is another matter entirely. If you've already enabled API caching (/guides/api-caching/enabling-api-caching) and implemented API cache control (/guides/api-caching/implementing-api-cache-control), you've probably run into this problem, which was aptly described by Phil Karlton (http://martinfowler.com/bliki/TwoHardThings.html):

> *There are only two hard things in computer science: cache invalidation and naming things.*

This guide explains how to use the Fastly API to purge your API cache with surrogate keys (/guides/purging/getting-started-with-surrogate-keys). Surrogate keys allow you to reduce the complexity of caching an API by combining multiple cache purges into a single key-based purge.

What's a surrogate key?

Surrogate keys allow you to selectively purge related content. Using the `Surrogate-Key` header, you can "tag" an object, such as an image or a blog post, with one or more keys. When Fastly fetches an object from your origin server, we check to see if you've specified a `Surrogate-Key` header. If you have, we add the response to a list we've set aside for each of the keys.

When you want to purge all of the responses associated with a key, issue a key purge (/guides/purging/single-purges#purging-with-keys) request and all of the objects associated with that key will be purged. This makes it possible to combine many purges into a single request. Ultimately, it makes it easier to manage categorically related data.

To learn more about surrogate keys and to see how you can integrate them into your application, see our guide on getting started with surrogate keys (/guides/purging/getting-started-with-surrogate-keys).

Example: Purging categories

To see how surrogate keys work in conjunction with an API endpoint, imagine you have an online store and an API endpoint that returns the details of a product. When a user wants to get information about a specific product, like a keyboard, the request might look like this:

```
GET /product/12345
```

If your API is using Fastly and the response is not already cached, Fastly will make a request to your API's origin server and receive a response like this:

```
HTTP/1.1 200 OK
Content-Type: text/json
Cache-Control: private
Surrogate-Control: max-age=86400
Surrogate-Key: peripherals keyboards

{id: 12345, name: "Uber Keyboard", price: "$124.99"}
```

You knew that entire product categories would occasionally need to be purged, so you thoughtfully included the `peripherals` and `keyboards` product categories as keys in the `Surrogate-Key` header. When Fastly receives a response like this, we add it to an internal map,

strip out the `Surrogate-Key` header, cache the response, and then deliver it to the end user.

Now imagine that your company decides to apply a 10% discount to all peripherals. You could issue the following key purge to invalidate all objects tagged with the `peripherals` surrogate key:

```
PURGE /service/:service_id/peripherals
```

When Fastly receives this request, we reference the list of content associated with the `peripherals` surrogate key and systematically purge every piece of content in the list.

Relational dependencies

Your API can use surrogate keys to group large numbers of items that may eventually need to be purged at the same time. Consider the example presented above. The API for your online store could have surrogate keys for product types, specific sales, or manufacturers.

From this perspective, the `Surrogate-Key` header provides Fastly with information about relations and possible dependencies between different API endpoints. Wherever there's a relation between two different types of resources in an API, there might be a good reason to keep them categorized by using a surrogate key.

Example: Purging product reviews and action shots

To learn how surrogate keys can help with relational dependencies, imagine that your online store wants to allow buyers to post product reviews and "action shots" depicting the products in use. To support these new features, you'll need to change your API. First, you'll need to create a new `review` endpoint:

```
GET /review/:id
POST /review
```

Next, you'll need to create a new `action_shot` endpoint:

```
POST /product/:id/action_shot
GET /product/:id/action_shot/:shot_id
```

Since both of the new endpoints refer to specific products, they'll need to be purged when relevant product information changes. Surrogate keys are a perfect fit for this use case. You can implement them by modifying the `review` and `action_shot` to return the following header:

```
Surrogate-Key: product/:id
```

This relates each of the endpoints to a specific product in the cache (where `:id` the product's unique identifier). When the product information changes, your API issues the following a key purge:

```
PURGE /service/:service_id/product/:id
```

When Fastly receives this request, we purge each of the related endpoints at the same time.

Variations on a theme

You'll also want to consider using surrogate keys if your API has many different endpoints that all derive from a single source. Any time the source data changes, each of the endpoints associated with it will need to be purged from the cache. By associating each of the endpoints with a surrogate key, a single purge can be issued to purge them from the cache when the source changes.

Example: Purging product images

To understand how this works, imagine that your online store has an API endpoint for retrieving product images in various sizes:

```
GET /product/:id/image/:size
```

This endpoint returns an image of the appropriate `:size` (e.g., `small`, `medium`, `large`) for the product of the given `:id`. To save disk space, you opt to have the API generate each specifically sized image from a single source image using an imaging library like ImageMagick (http://www.imagemagick.org/). Since the sales and marketing team uses the API to upload new product images, you set up the endpoint to include a surrogate key:

```
Surrogate-Key: product/:id/image
```

When the PUT endpoint for uploading a product image is called, the API sends the following purge request:

```
PURGE /service/:service_id/product/:id/image
```

When Fastly receives this request, we purge all size variations of the product image.

- Guides (/guides/) > Developer's tools > Conditions (/guides/conditions/)

§ About conditions (/guides/conditions/about-conditions)

Conditions control how requests are processed. You can use them to add logic to any basic configuration object in a service and have them control if and when that object is applied. Conditions require minimal programming. They allow you to wrap configuration objects attached to your service in a VCL (/guides/vcl/guide-to-vcl) IF statement.

Before you start using conditions

Be sure you understand the construction of basic logical expressions before you start using conditions. Specifically, you should understand basic C-style logical expression syntax (e.g., basic logic, operators such as && and precedence) when working with conditions. A basic programming guide that deals with "IF" style expressions in either the C or Perl language (the Tizag Perl tutorial (http://www.tizag.com/perlT/) is a good one to start with). Even though they aren't directly applicable to our condition examples (/guides/conditions/using-conditions), the syntax of these languages is similar to VCL.

A simple condition example

The simplest way to explain how Fastly handles conditions is this IF statement:

```
IF
   this condition happens
THEN
   respond this way
```

A practical example can demonstrate this. The vast majority of the time, your site processes requests for information as usual, but every so often customers mistype a search term or simply can't find what they're looking for and you're forced to display a 404 Not Found error. You've realized that when that happens, the standard 404 Not Found errors on your website aren't as helpful as they could be. To fix this, any time your server can't find what a customer is looking for (a condition), you want to display a customized 404 message instructing customers to contact your support team for help (a response).

In plain English, the IF statement might look like this:

```
IF
   404 Not Found is what we have to tell the customer
THEN
   respond with the special Contact Support page
```

The IF line in the example above is the *condition* you've set. The THEN line describes what will happen if that condition is met.

If you were to replace the English in the example above with VCL variables and a little bit of HTML, it might look like this instead:

```
IF
  beresp.status == 404
THEN
  respond with <html><body><h1>Can't find it?</h1><p>Contact support@example.com for help.</p>
</body></html>
```

Interested in doing this? We have step-by-step instructions for creating error pages with custom responses (/guides/basic-configuration/creating-error-pages-with-custom-responses).

Ideas for using conditions

Need some more ideas for when you could use conditions? Explore these:

Condition	Response	Learn how
A web robot wants to crawl a particular area of your website	Provide a customized robots.txt file defining which areas of your website should not be processed or scanned	Creating and customizing a robots.txt file (/guides/basic-configuration/creating-and-customizing-a-robots-file)
Your server needs to return a 404 Not Found response	Change the default caching time for only 404 responses from 3600 seconds (60 minutes) to 120 seconds (2 minutes)	Overriding caching defaults based on a backend response (/guides/basic-configuration/overriding-caching-defaults-based-on-a-backend-response)
Users request a popular page on your site but it's been moved to a different area	Have Fastly redirect the page requests at the edge, without having to go back to your origin server for it	Generating HTTP redirects at the edge (/guides/performance-tuning/generating-http-redirects-at-the-edge)

Types of conditions and when you can use them

We group conditions into three types:

- request conditions
- response conditions
- cache conditions

A condition's type dictates which configuration objects it can be applied to during a specific stage of the caching process (/guides/basic-concepts/how-caching-and-cdns-work). In addition, each stage of caching works with a different set of VCL variables (https://www.varnish-cache.org/docs/2.1/reference/vcl.html#variables) that can be used to create conditions.

Condition type	Applied when Fastly …	Works with which VCL variables

Request	processes a request	`client.*`
		`server.*`
		`req.*`

Response	processes a response to a request	`client.*`
		`server.*`
		`req.*`
		`resp.*`

Cache	receives a response from your origin, just before that response is (potentially) cached	`client.*`
		`server.*`
		`req.*`
		`beresp.*`

Where to go for more information

The Varnish Cache documentation (https://www.varnish-cache.org/docs/2.1/reference/vcl.html#variables) provides a complete list of variables you can use to craft conditions (/guides/conditions/using-conditions). Keep in mind, however, some of the variables Varnish allows may not be available or may have no meaning in the context of Fastly services. For more information, see our Guide to VCL (/guides/vcl/guide-to-vcl).

---

§ Troubleshooting conditions (/guides/conditions/troubleshooting-conditions)

If you are having problems using conditions, here are some common mistakes to look for:

- **Don't use the wrong case.** Varnish regular expressions are case sensitive.
- **Don't escape forward slashes.** Forward slashes don't need to be escaped in Varnish regular expressions.
- **Don't put the `if ()` statement in the Apply if field of the condition window.** The actual if statement is implied. You only need to type an evaluated expression. For example: `req.url ~ "^/special/"`. Most problems with conditions occur in the Apply if parameter because it

uses logical expressions to represent actual VCL variables that specify when a condition should be applied to a configuration object.

- **Don't use the `!~` (inverse regex match) to build regular expressions that exclude particular URLs.** For example, if you want to apply something to all URLs except those that start with `/admin`, the condition for this is `req.url !~ "^/admin"` and is entered in the Apply If field. An alternative to this expression would be `!(req.url ~ "^/admin")`.

Our cheatsheet (/guides/vcl/vcl-regular-expression-cheat-sheet) provides additional examples of using VCL with regular expressions.

---

§ Using conditions (/guides/conditions/using-conditions)

Conditions use the Varnish Configuration Language (VCL) (/guides/vcl/guide-to-vcl#about-varnish-and-why-fastly-uses-it) to define when an configuration object should be applied while processing requests to a cache server. Once you understand some basics about conditions (/guides/conditions/about-conditions), use this guide to learn about how to create conditions using the Fastly web interface and when to use them.

Where to find conditions

Conditions appear in two areas of Fastly's web interface:

- The Manage conditions page lists all conditions available to your configuration settings.
- Each configuration object displays conditions specifically attached to them.

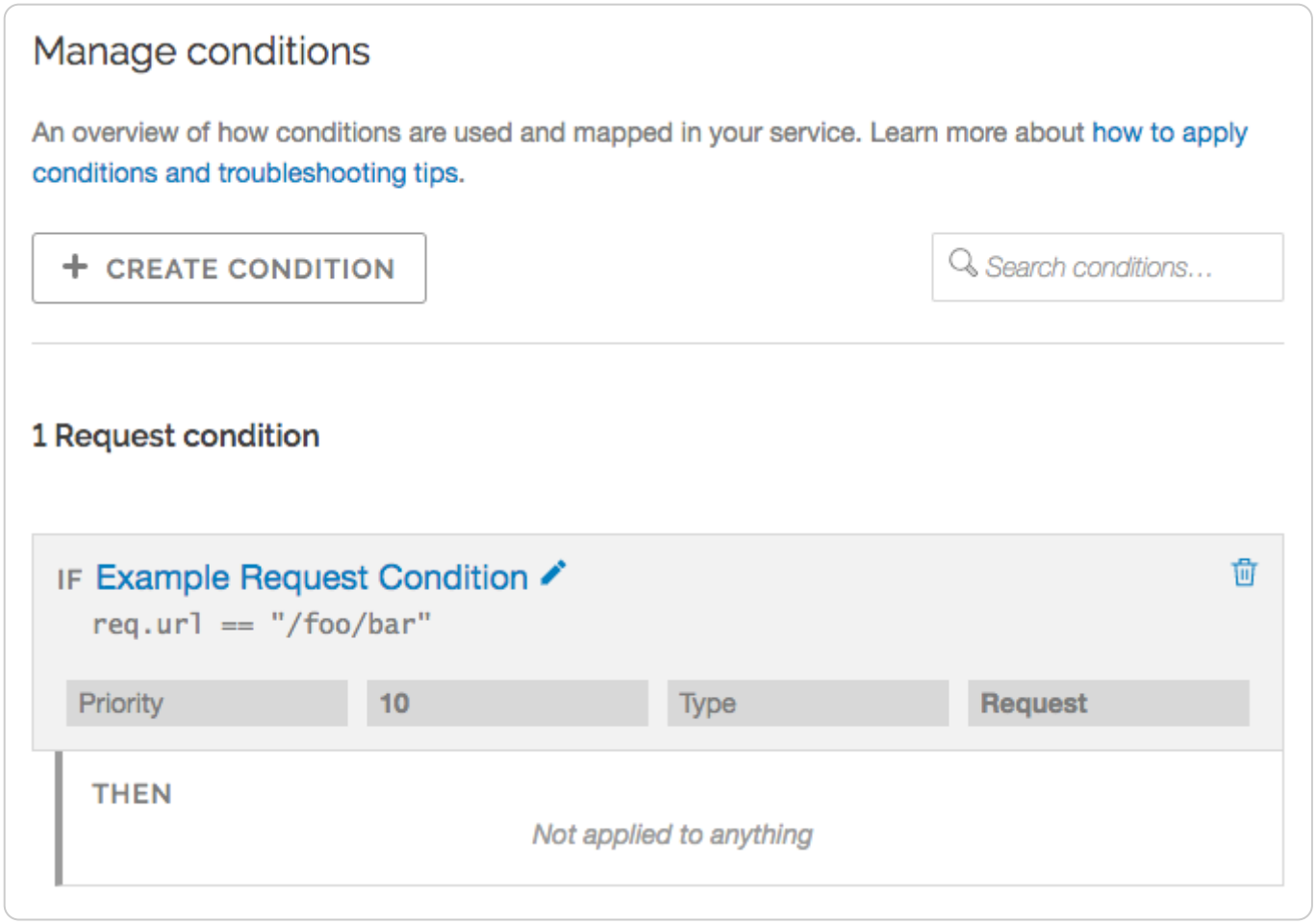
Conditions on the Manage conditions page

The Manage conditions page provides an overview of all the conditions currently available to your service. You can see at a glance which conditions are mapped to configuration objects. It allows you to create new conditions and search for existing ones.

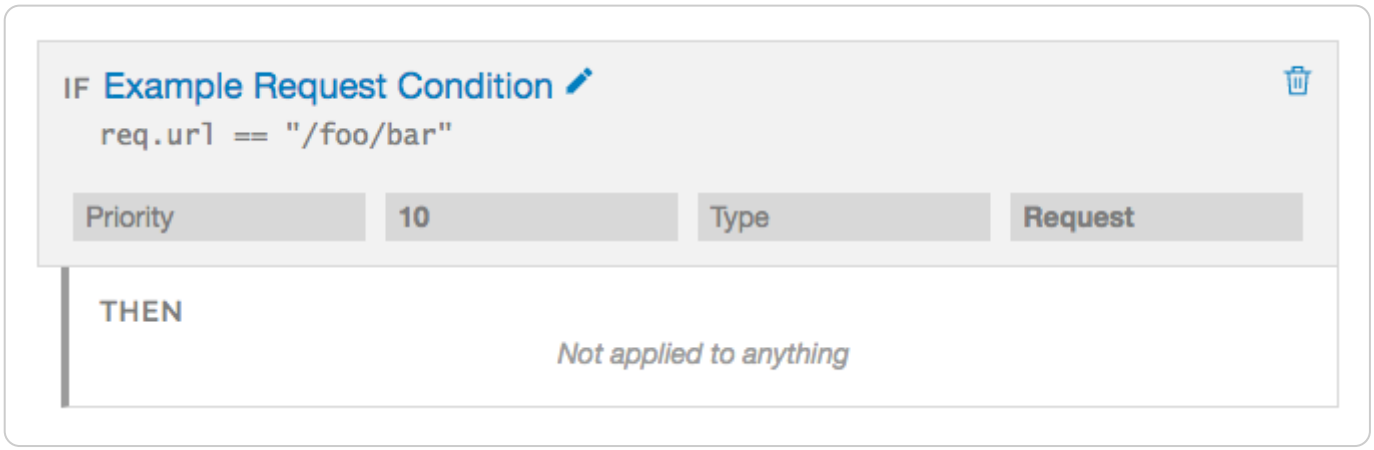
To view conditions on the Manage conditions page:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Conditions** link. A list of all conditions for your service appears.

For example, this service has one request condition available:

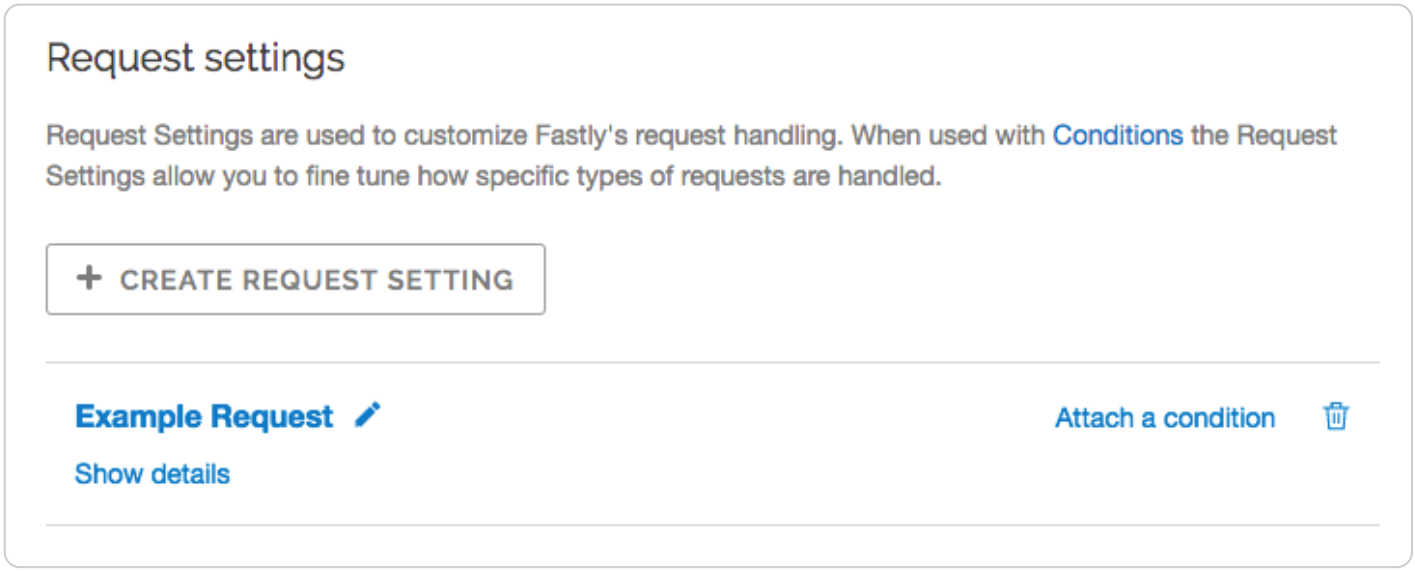


The Example Request Condition shown above currently isn't applied to a configuration object (as indicated by "Not applied to anything"). If it was, it would instead appear similar to this:

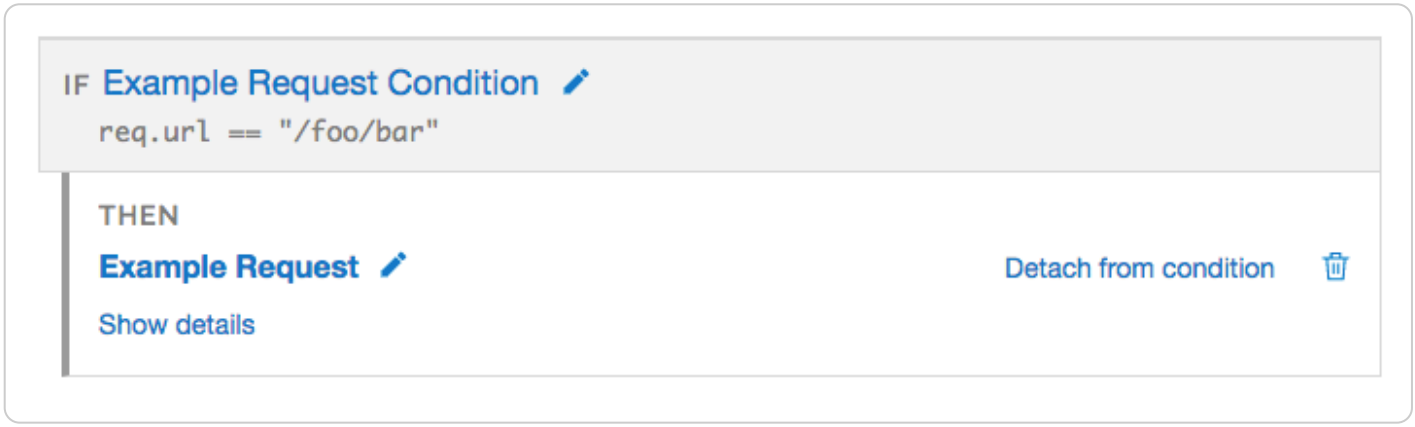


Conditions attached directly to a configuration object

Configuration objects appear differently in the web interface when conditions are attached to them. For example, this request setting has no condition attached to it:



Once you click the Attach a condition link to create a new condition or attach an existing condition, however, the web interface changes how the configuration object appears:



By default, configuration objects hide the majority of details for any attached conditions. You can unhide those details by clicking the Show details link. When expanded, the details vary depending on the type of condition.

Parts of a properly configured condition

Conditions require only a few parameters, making them appear deceptively simple. Specifically, they require:

- a **Type** parameter that classifies the condition being added. If added via the Manage conditions page, the type can always be manually selected. If added via the Attach a condition link on a configuration object, the type is automatically applied whenever possible.
- a **Name** parameter that serves human-readable identifier of the condition.
- an **Apply if** statement containing the logical expression to execute in VCL to determine if condition resolves as True or False.

Most problems with conditions (/guides/conditions/troubleshooting-conditions) occur in the Apply if parameter.

Performing matches on basic logical expressions

Properly configured conditions can perform matches on complicated logical expressions specified in the Apply if parameter. For example:

This logical expression …	Matches when …
`client.ip == "127.0.0.1"`	The client requesting a resource on your service has the IP `127.0.0.1`.
`req.http.host == "example.com"`	The host header of the incoming request is `example.com`.
`req.request == "POST" && req.url ~ "^/api/articles/"`	The request is a POST and the URL begins with `/api/articles/`.

The `client.ip`, `req.http.host`, `req.request`, and `req.url` conditions shown above all represent configuration variables in VCL (/guides/vcl/).

Using operators to perform matches on complex logical expressions

You could also get creative and create a more complex condition used by Fastly that might have an Apply if parameter that looked like this:

```
req.http.host == "www.example.com" && (req.url !~ "^/foo" || req.url !~ "^/bar/" || req.url !~ "^/baz/")
```

This condition tells the cache server that the URL should equal `www.example.com` and the URL cannot match `www.example.com/foo` or `www.example.com/bar` or `www.example.com/baz`. You might use this type of condition when you have multiple variables or options and want to fine-tune your results. In this example, you are indicating that you don't want URLs that contain `foo`, `bar`, or `baz` by using the following operators:

This operator ...	Does this ...
`( )`	groups expressions and restricts alteration to part of the regex
`||`	performs an alternation where each variable is checked until it finds a variable that is true
`!~`	excludes any URLs that include the specified variables

An example of adding conditions

The scenario: You want to add a new origin server that handles a specific portion of your API requests. Some requests to this API must be cached differently than other requests to your API, so you want to set special headers for specific types of requests. Specifically, you don't want your new origin server to cache PUT, POST, or DELETE requests because they're special for this particular API and send back extra, time dependent, meta-information in each response. And finally, you want to track the effectiveness of doing this. To accomplish all of this using conditions via the Fastly web interface, you would:

1. Create a new origin server to handle the special API traffic.
2. Create a new condition that tells the cache how to route some of the API requests to that origin server.
3. Create a new cache setting object to ensure the origin server caches only the correct responses.
4. Create a new condition that specifies when the cache settings object should be applied.
5. Create a new header to track the specific type of API requests.
6. Create a new request condition to make sure that the header is only set on specific type of request.
7. Check your work.

Create a new origin server

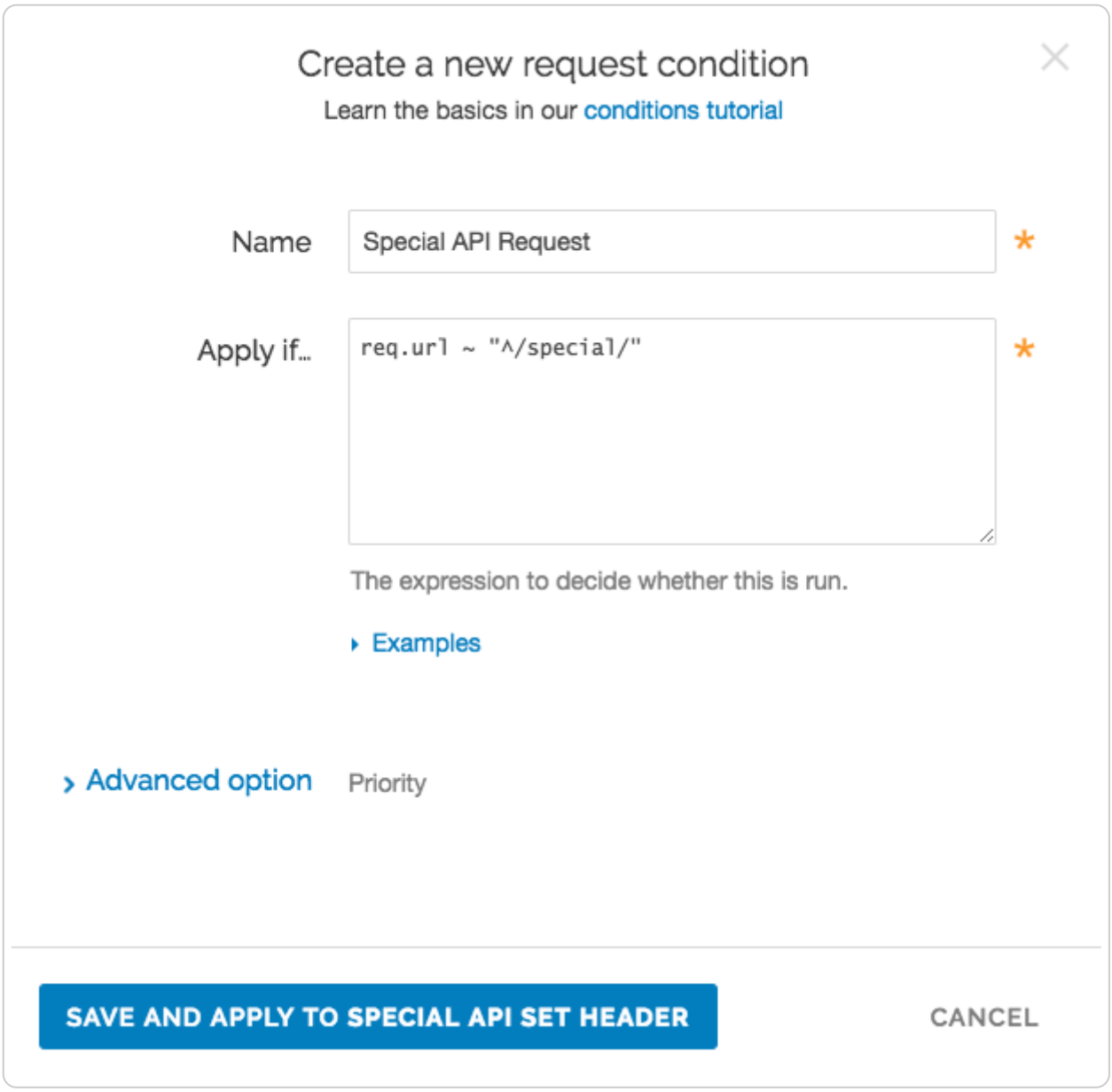
To create a new origin server that will handle the special API traffic, follow the instructions for connecting to origins (/guides/basic-configuration/connecting-to-origins). You'll add specific details about your API server when you fill out the **Create a host** fields:

- In the **Name** field, type a name for your API server (for example, `Special API Set Header`).
- In the **Address** field, type the IP address (or hostname) of the API server.

Create a request condition

Once you've created a new origin server to handle the special API traffic, tell the cache how to route requests to this origin server by creating a request condition.

1. In the **Hosts** area, click the **Attach a condition** link next to the name of the origin server you just created. The Add a condition to window appears.
2. You can either select an available condition or you can click the **Create a new request condition** button. The Create a new request condition window appears.

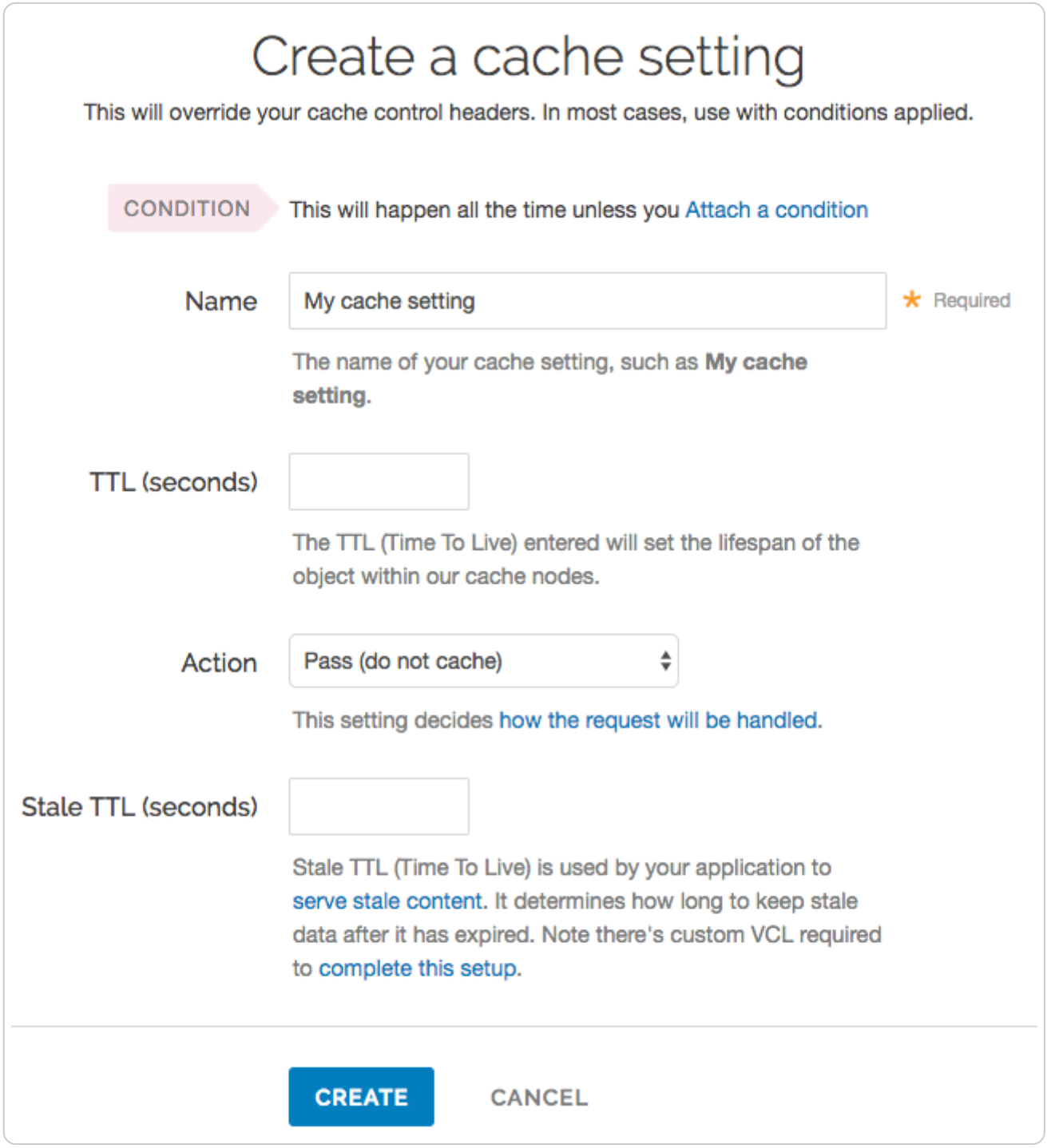


3. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example `Special API Request`).
   - In the **Apply if** field, type the appropriate request condition that will be applied (for example, `req.url ~ "^/special/"` could address all requests related to the special API server).
4. Click the **Save and apply to** button to create the new condition for the host.

Create a cache settings object

Requests are now are being properly routed to the new origin server. Next, create a cache settings object to ensure the origin doesn't cache any responses from PUT, POST, or DELETE requests. They're special for this particular API and send back extra, time dependent, meta-information in each response.

1. Click the **Settings** link. The Settings page appears.
2. In the **Cache Settings** area, click the **Create cache setting** button. The Create a cache setting page appears.



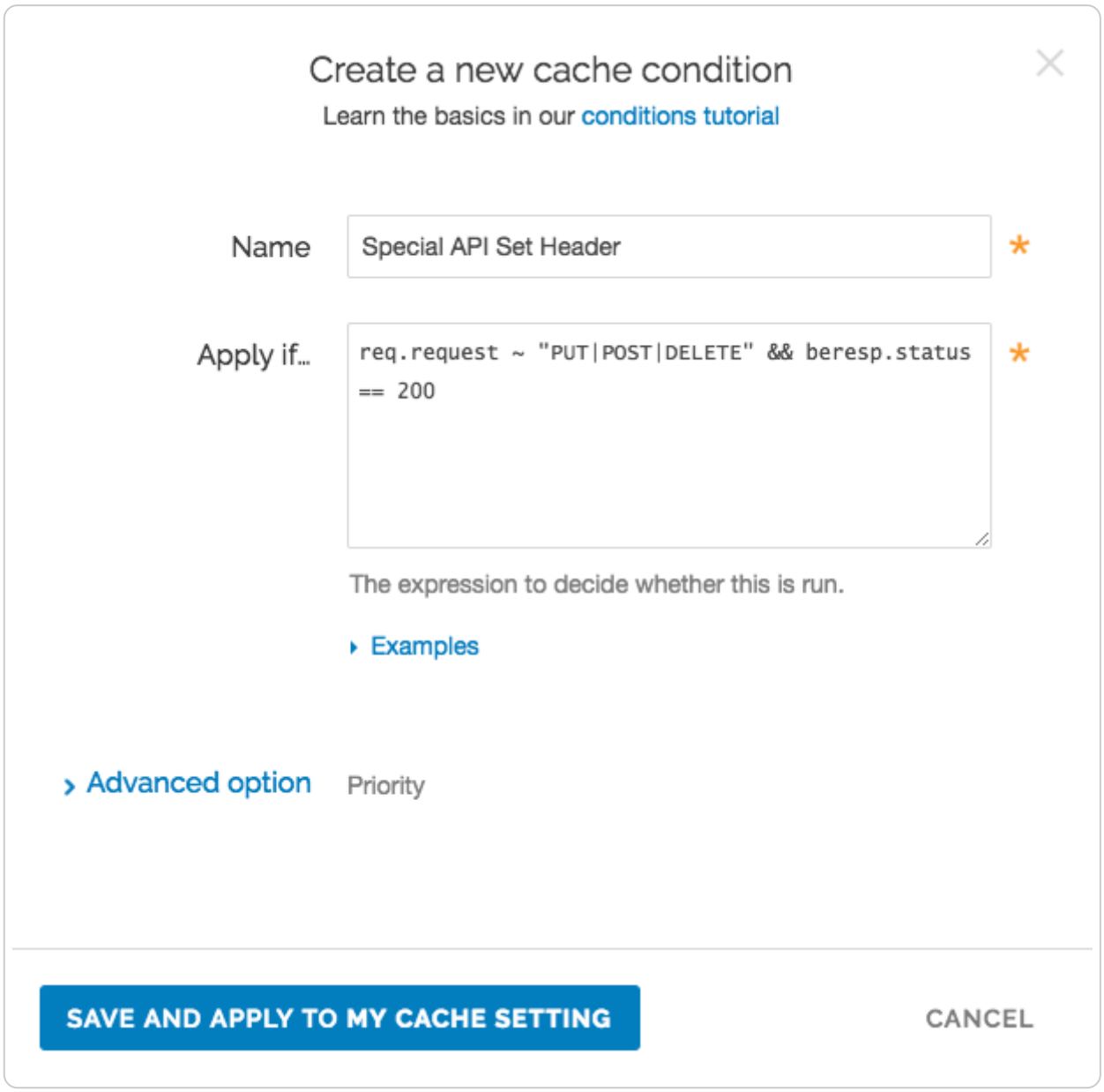
3. Fill out the **Create a cache setting** fields as follow:
   - In the **Name** field, type a descriptive name for the new cache settings.
   - Leave the **TTL (seconds)** field set to its default value.
   - From the **Action** menu, select **Pass (do not cache)**.
   - Leave the **Stale TTL (seconds)** field set to its default value.

4. Click the **Create** button.

Create and apply a condition to the cache settings object

Create a new condition that specifies when the cache settings object should be applied.

1. In the **Cache Settings** area, click the **Attach a condition** link next to the name of the cache setting you just created. The Add a condition to window appears.
2. Click **Create a new cache condition** button. The Create a new cache condition window appears.



3. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Special API Set Header`).

- In the **Apply if** field, type the appropriate request condition that will be applied (for example, `req.request ~ "PUT|POST|DELETE" && beresp.status == 200`).

4. Click the **Save and apply to** button to create the new condition for the cache setting.

Create a new header

To make sure you can track the effectiveness the new API, create a new header so you can use it to gather information about the special API requests as they happen.

1. Click the **Content** link. The Content page appears.
2. In the **Headers** area, click the **Create header** button to create a new header. The Create a header page appears.

Create a header

Learn more about this section in our headers tutorial.

CONDITION This will happen all the time unless you Attach a condition

Name

Special API Set Header

* Required

The name of your header, such as **My header**.

Type / Action

Response | Set

The type of header and the action performed on it.

Destination

http.super

* Required

The name of the header that will be affected by the selected action. For example: **http.Content-Type**, **http.Set-Cookie**, **http.Via**, **http.Location**, or **http.Access-Control-Allow-Origin**.

Source

"Thanks for asking"

* Required

New content for the header. Can be a static value (e.g. string or number) or a dynamic value (e.g. existing header or a GeoIP value). Please use quotes for string values.

Ignore if set

No

If switched to **Yes**, the action will not be performed if the header in Destination exists.

Priority

10

* Required

The order in which the header rules execute within the condition.

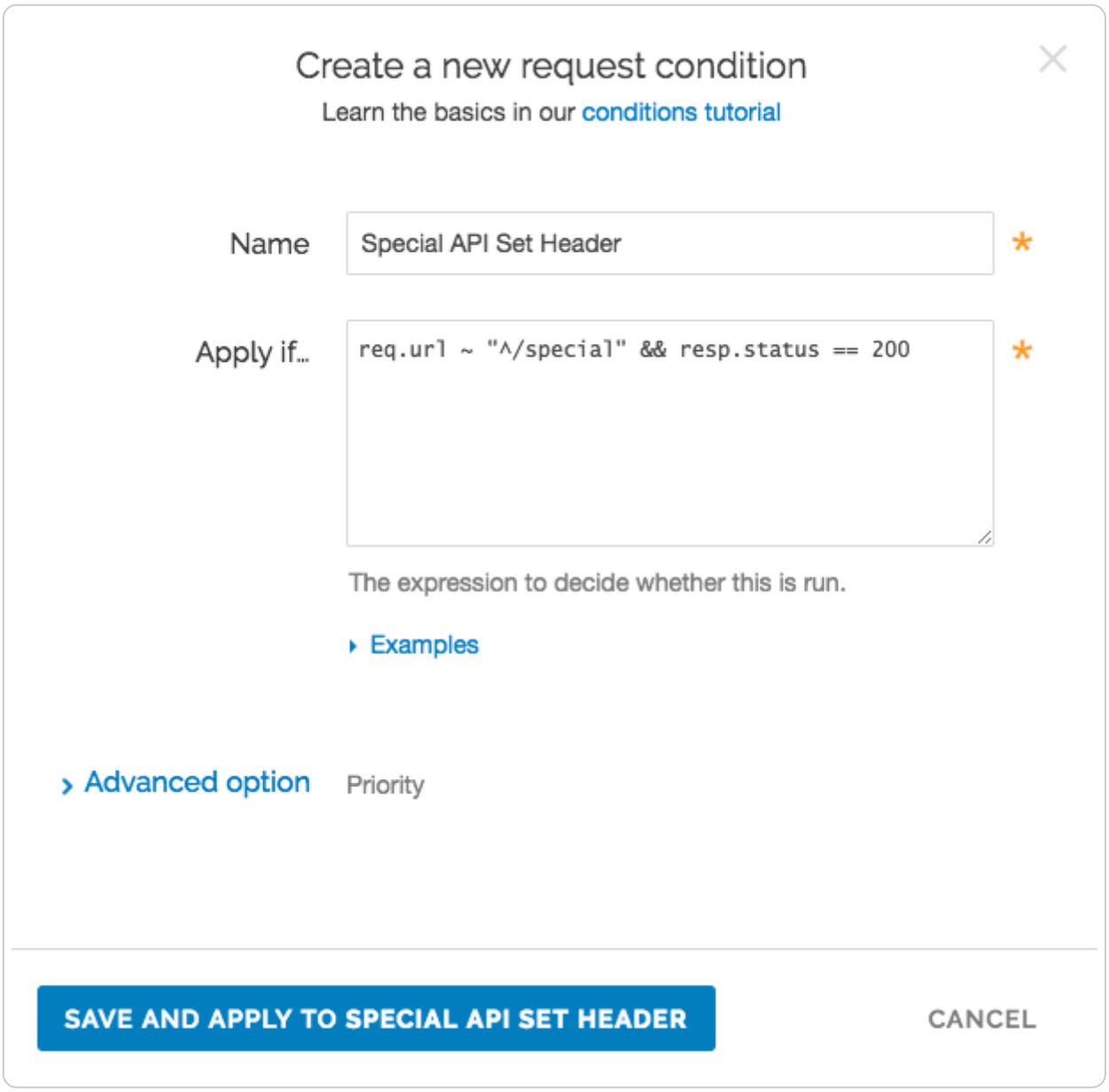
CREATE CANCEL

3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the new header (for example, `Special API Set Header`).
   - From the **Type** menu, select **Response** and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header that will be affected by the action (for example, `http.super`).
   - In the **Source** field, type a description of the source where the content for this header comes from (for example, `"Thanks for asking!"`).
   - Leave the **Ignore if set** and **Priority** fields set to their default settings.
4. Click **Create**.

Create a request condition for the new header

Once the header is created, create an associated condition to ensure this header is only set on that special type of request.

1. In the **Headers** area, click the **Attach a condition** link next to the name of the new header you just created. The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Special API Response Condition`).
   - In the **Apply if** field, type the appropriate request condition that will be applied (for example, `req.url ~ "^/special" && resp.status == 200`).
3. Click the **Save and apply to** button to create the new condition for the header.

Check your work

Before activating the configuration, review the generated VCL (/guides/vcl/previewing-and-testing-vcl-before-activating-it) to see how Fastly converted the objects and conditions into actual VCL. For the example shown above, the VCL for the request condition appears as:

```
# Condition: Special API Request Prio: 10
if (req.url ~ "^/special/") {
  set req.backend = F_Special_API_Server;
}
#end condition
```

The cache settings and condition VCL appears as:

```
if (req.request ~ "PUT|POST|DELETE" && beresp.status == 200) {
  set beresp.ttl = 0s;
  set beresp.grace = 0s;
  return(pass);
}
```

And the new header response condition VCL appears as:

```
# Condition Special API Response Condition Prio: 10
if (req.url ~ "^/special" && resp.status == 200) {

  # Header rewrite Special API Set Header: 10
  set resp.http.super = "Thanks for asking!";
}
```

As you become more familiar with the VCL syntax and programming, look at the generated VCL to see if the configuration is doing what you think it is doing (most VCL is pretty simple once you know what the variables are referring to).

- Guides (/guides/) > Developer's tools > Edge Dictionaries (/guides/edge-dictionaries/)

§ About Edge Dictionaries (/guides/edge-dictionaries/about-edge-dictionaries)

Fastly offers instantly updatable, global, Edge Dictionaries for use with your services (/guides/basic-setup/working-with-services).

Edge Dictionaries simplify services

Edge Dictionaries are made up of dictionary containers and the dictionary items within them. In combination, containers and items allow you to store data as key-value pairs and turn frequently repeated statements like this:

```
if (something == "value1") {
  set other = "result1";
} else if (something == "value2") {
  set other = "result2";
}
```

into a single function that acts as constant:

```
table <ID> {
  "KEY_STRING": "VALUE_STRING",
  "KEY_STRING2": "VALUE_STRING2",
  ...
}
```

Once you attach a dictionary container to a version of your service with the API and that service is activated, the data in it becomes "versionless." This means you can add to and update (/guides/edge-dictionaries/creating-and-manipulating-dictionary-items) the data an Edge Dictionary contains using a single API call at any time after it is created, without ever incrementing a service's version. You can perform lookups on the dictionary items in an Edge Dictionary using functions like `table.lookup(<ID>, "KEY_STRING")` or `table.lookup(<ID>, "KEY_STRING", "DEFAULT_VALUE_STRING")` in your configuration.

When Edge Dictionaries might be useful

- Content sharing and social media outlets updating large referer blacklists
- Mobile advertisers validating a key to prevent cache-bust guessing
- Customers authenticating valid user keys at the edge (see our guide on private Edge Dictionaries (/guides/edge-dictionaries/private-dictionaries))
- Global publishers redirecting users to a specific country site based on geo-location
- Image providers performing token checks for certain objects
- Advertising technology companies blocking bad actors at edge
- Customers deploying user interface versions with simple value change via API

Limitations and considerations

When creating Edge Dictionaries (/guides/edge-dictionaries/creating-and-using-dictionaries), keep the following things in mind as you develop your service configurations:

- **Edge Dictionaries created with custom VCL cannot be manipulated using the API.** If you create a dictionary container using the API (/api/config#dictionary), you can use the API to make changes to it, and use custom VCL to interact with it. If you create a dictionary container using custom VCL, that dictionary must always be manipulated via custom VCL.

- **Dictionary containers are limited to 1000 dictionary items.** If you find your dictionary containers approaching this item limit, contact us (mailto:support@fastly.com). We may be able to help you figure out an even more efficient way to do things with your Edge Dictionaries.
- **Dictionary item keys are limited 256 characters and values are limited to 8000 characters.** Dictionary items are made up of key-value pairs with character limits. Be sure to take this into account when designing your Edge Dictionaries.
- **Dictionary item keys are case sensitive.** Dictionary item names are case sensitive. Be sure to take this into account when designing your Edge Dictionaries.
- **Event logs don't exist for Edge Dictionary changes.** If you add, update, or remove a dictionary item, there will be no record of it. The only record of a change will exist when you compare service versions (/guides/basic-setup/working-with-services#comparing-different-service-versions) to view the point at which the dictionary container was associated with the service version in the first place.
- **When you delete a dictionary container, you'll only delete it from the service version you're editing.** Dictionary containers are tied to versions and can be cloned and reverted. When using Edge Dictionaries, we want you to be able to do things like delete a dictionary container from a current version of your service in order to roll back your configuration to a previous version using as few steps as possible.
- **When you delete a dictionary container, we don't delete the dictionary items inside it.** The dictionary items in a dictionary container are versionless. When you change service versions, we want you to still be able to access the data.
- **Dictionary item deletions are permanent.** Because we don't store data, if you delete a dictionary item, the entry is gone forever from all service versions.

§ Creating and manipulating dictionary items (/guides/edge-dictionaries/creating-and-manipulating-dictionary-items)

A dictionary item is a key-value pair that makes up an entry in a dictionary container in an Edge Dictionary. Once you create an Edge Dictionary (/guides/edge-dictionaries/creating-and-using-dictionaries) and associate the dictionary container with a service, any dictionary items created will appear in your generated VCL.

For example, if you were using Edge Dictionaries to control geolocation redirects, the table would appear similar to this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
}
```

If you already have a dictionary container associated with an active version of your service, you can easily add, update, or delete the items in it as long as you know the `dictionary_id`.

In our geolocation example, you would find your `dictionary_id` using the following API call:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN'
https://api.fastly.com/service/<service_id>/version/<version_number>/dictionary/geoip_redirect
```

which would return this response:

```
{
  "version": <version_number>,
  "name": "geoip_redirect",
  "id": "<dictionary_id>",
  "service_id": "<service_id>"
}
```

Adding new items to dictionary

You can add new dictionary items without having to increment your service version number. For example, this API call to a geolocation table to add a new dictionary item:

```
curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_key=NZ&item_value=www.example.com.au'
"https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/item"
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "service_id": "<service_id>",
  "item_key": "NZ",
  "item_value": "www.example.com.au"
}
```

The table in the generated VCL would then be updated with the new dictionary item and look like this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
  "NZ"  :  "www.example.com.au",
}
```

Listing dictionary items

You can view all of the dictionary items in an Edge Dictionary. For example, this API call:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/dictionary/<
dictionary_id>/items
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "service_id": "<service_id>",
  "item_key": "some_key",
  "item_value": "some_value",
  "created_at": "2016-04-21T18:14:32+00:00",
  "deleted_at": null,
  "updated_at": "2016-04-21T18:14:32+00:00"
}
```

Upserting dictionary items

You can create and update dictionary items regardless of whether or not they exist. For example, the following API call to the geolocation table to update an existing dictionary item or create it if it doesn't exist:

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_value=www.example.co.aq' "https://api.fa
stly.com/service/<service_id>/dictionary/<dictionary_id>/item/AQ"
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "item_key": "AQ",
  "item_value": "www.example.co.aq",
  "service_id": "<service_id>"
}
```

The table in the generated VCL would then be updated with the new dictionary item and look like this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
  "NZ"  :  "www.example.com.au",
  "AQ"  :  "www.example.co.aq",
}
```

Updating dictionary items

You can also update any dictionary item without having to increment your service version number. For example, the following API call to the geolocation table to update an existing dictionary item:

```
curl -X PATCH -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_value=www.example.co.uk' "https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/item/NZ"
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "item_key": "NZ",
  "item_value": "www.example.co.uk",
  "service_id": "<service_id>"
}
```

The table in the generated VCL would then be updated with the new dictionary item and look like this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
  "NZ"  :  "www.example.co.uk",
  "AQ"  :  "www.example.co.aq",
}
```

Batch updating dictionary items

You can update up to 1,000 dictionary items with a single API call. The following actions are available within a batch update:

- **Upsert** - Creates an item if it doesn't exist, otherwise modifies the existing one.
- **Create** - Creates a new item, but will not update an existing one.
- **Update** - Updates an existing item, but will not create a new one if it doesn't exist.

- **Delete** - Permanently deletes the item from the dictionary.

For example, to batch update existing dictionary items in the geolocation table, create a new file called `batch.json` that contains the following JSON-encoded data:

```
{
  "items": [
    {
      "op": "create",
      "item_key": "JP",
      "item_value": "www.example.co.jp"
    },
    {
      "op": "update",
      "item_key": "GB",
      "item_value": "www.example.co.uk"
    },
    {
      "op": "delete",
      "item_key": "IT"
    }
  ]
}
```

Now you can make the following API call:

```
curl -X PATCH -H 'Content-Type: application/json' -H 'Fastly-Key: FASTLY_API_TOKEN' -d @batch.j
son "https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/items"
```

See the API documentation (/api/config#dictionary_item_dc826ce1255a7c42bc48eb204eed8f7f) for more information.

Deleting a dictionary item

⚠ **WARNING:** Dictionary item deletions are permanent. Fastly does not store data. If you delete a dictionary item, the entry is gone forever from all versions of your service.

To remove an item from your table, use this API call:

```
curl -X DELETE -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<service_id>/di
ctionary/<dictionary_id>/item/NZ
```

Unlike creation and update of dictionary items, the API call returns no response.

§ Creating and using Edge Dictionaries (/guides/edge-dictionaries/creating-and-using-dictionaries)

Edge Dictionaries (/guides/edge-dictionaries/about-edge-dictionaries) allow you to create logic that doesn't need to be attached to a configuration service version. Edge Dictionaries are made up of dictionary containers and dictionary items. You can use dictionary items to create and store key-value pairs. Attaching dictionary containers to a service version allows you to turn frequently repeated statements into single function statements that act as a constant.

To create an Edge Dictionary and use it within your service you need to:

1. Create an empty dictionary container in a working service on a version that's unlocked and not yet activated.
2. Activate the new version of the service you associated with the empty dictionary container.
3. Add dictionary items to the newly created dictionary container.

Once the dictionary container is created, properly associated, and filled with dictionary items, it can be called in your service.

For example, say you have a referer blacklist that changes frequently and you want to associate it with a service. Any time that service's configuration changes, especially if the configuration rolls back to a previous version, you would want the blacklisted referer domains to continue to remain with the service configuration instead of being removed. Edge Dictionaries would help you do this.

> ★ **TIP:** You can create a private Edge Dictionary (/guides/edge-dictionaries/private-dictionaries) to store dictionary items that can't be listed or read.

Create an empty dictionary container within a service

In order to use a dictionary container, start by creating an empty one within an unlocked version of a service.

Before an Edge Dictionary can be manipulated, its dictionary container must be associated with at least one service version that is not locked and not active so that the service becomes aware of the dictionary's existence.

For example, if you were creating a referer blacklist via the API, you would make an API call by running this command:

```
curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'name=referer_blacklist' https://api.fastly.com/service/<service_id>/version/<version_number>/dictionary
```

which would return:

```
{
  "created_at": "2017-05-03T16:11:41+00:00",
  "deleted_at": null,
  "id": "<dictionary_id>",
  "name": "referer_blacklist",
  "service_id": "<service_id>",
  "updated_at": "2017-05-03T16:11:41+00:00",
  "version": <version_number>,
  "write_only": false
}
```

Activate the service associated with the dictionary container

In order for an Edge Dictionary to appear in generated VCL so it can be referred to later, the version associated with the dictionary container must be activated.

In our referer blacklist example, you would make this API call to activate service version associated with the empty dictionary container you created:

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<service_id>/version/<version_number>/activate
```

The response would be this:

```
{
  "number": <version_number>,
  "active": true,
  "service_id": "<service_id>"
}
```

Add dictionary items

Once the dictionary container becomes associated with the configuration of a service, you can begin populating it with dictionary items.

For example, you would use the following API call for each URL you want to add a URLs to your referer blacklist:

```
curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_key=example-referer.org&item_value=true' "https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/item"
```

The response for each URL added would look similar to this:

```
{
  "dictionary_id": "<dictionary_id>",
  "service_id": "<service_id>",
  "item_key": "example-referer.org",
  "item_value": "true"
}
```

Once the blacklisted URLs are added as items in your dictionary container, you can find them in your generated VCL by looking for a table similar to this:

```
table referer_blacklist {
  "example-referer.org": "true",
  "another-referer.net": "true",
  "sample-referer.com": "true",
}
```

Using a service to call Edge Dictionaries

When you create Edge Dictionaries via API calls, the dictionary contents aren't tied to any single version of your service.

The logic needed to *interact* with the table of information the Edge Dictionary creates, however, is always tied to a service version.

For example, adding a new referer to your blacklist requires that you specifically interact with the Edge Dictionary at some point after you create it. You could do this via API calls because its data would not require a service version activation. The dictionary was created via API calls not via custom VCL.

Specifically, you would set the host of the referer to a header by including custom VCL like this:

```
// versioned vcl
sub vcl_recv {

  # capture host of referer into a header
  set req.http.Referer-Host = regsub(req.http.Referer, "^https?://?([^:/\s]+).*$", "\1");

  # check if referrer host is in blacklisted table
  if (table.lookup(referer_blacklist, req.http.Referer-Host)) {
    # ResponseObject: forbidden-referrer
    error 900 "Fastly Internal";
  }
  #end condition
}

sub vcl_error {

  if (obj.status == 900) {
    set obj.http.Content-Type = "";
    synthetic {""};
    return(deliver);
  }
}
```

Custom VCL examples

These examples illustrate how to use Edge Dictionaries in custom VCL. The dictionaries are created via API calls and are displayed in the tables.

Referer blacklist

This example returns a 403 error message if the referer is in the dictionary.

```
// dictionary items can be added, updated, removed via the API
// does not require cloning and activating versions
table bad_guys {
  "example.com" : "nope",
  "fastly.com" : "nope",
}
// versioned vcl
sub vcl_recv {
  set req.http.Referer-Host = regsub(req.http.Referer, "https?://([^/]+)/.*", "\1");
  set req.http.Referer-Check = table.lookup(bad_guys, req.http.Referer-Host, "yes");
  if (req.http.Referer-Check == "nope") {
    error 403;
  }
}
```

CORS origin database

This example adds the origins in the dictionary to the `Access-Control-Allow-Origin` header.

```
// dictionary items can be added, updated, removed via the API
// does not require cloning and activating versions
table acceptable_origins {
  "http://example.com" : "yes",
  "http://fastly.com" : "yes",
}
// versioned vcl
sub vcl_deliver {
  set req.http.CORS = table.lookup(acceptable_origins, req.http.Origin, "nope");
  if (req.http.CORS == "yes") {
    set resp.http.Access-Control-Allow-Origin = req.http.Origin;
  }
}
```

TTL database

This example sets the TTLs for the URLs in the dictionary.

```
// dictionary items can be added, updated, removed via the API
// does not require cloning and activating versions
table ttls {
  "/" : "60",
  "/public" : "86400",
  "/api" : "3600",
  "/foo" : "7200",
  "/user" : "5"
}
// versioned vcl
sub vcl_fetch {
  /* cut URL down to first directory, or just / */
  if (req.url.path ~ "^(/[^/\?]*)") {
    /* should always be true */
    set beresp.ttl = std.atoi(table.lookup(ttls, re.group.1, "30"));
  }
```

§ Private Edge Dictionaries (/guides/edge-dictionaries/private-dictionaries)

Private Edge Dictionaries store dictionary items that can't be listed or read via the API.

Limitations and considerations

When creating private Edge Dictionaries, keep the following things in mind:

- You can create, read, update, and delete a private dictionary

- You cannot update the `write_only` attribute of a dictionary
- You can create, update, and delete items that belong to a private dictionary
- You cannot view items that belong to a private dictionary via the API
- Depending on how your service (/guides/basic-setup/working-with-services) is configured, data stored in private Edge Dictionaries can be sent in headers (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) and to log streaming endpoints (/guides/streaming-logs/setting-up-remote-log-streaming).

⚠ **WARNING:** To edit or delete dictionary items in a private dictionary, you'll need to remember the keys of the dictionary items (/guides/edge-dictionaries/creating-and-manipulating-dictionary-items).

Creating a private dictionary container

To use a private dictionary container, start by creating an empty one within an unlocked version of a service.

Before a private Edge Dictionary can be manipulated, its private dictionary container must be associated with at least one service version that is not locked and not active so that the service becomes aware of the private dictionary's existence.

For example, if you were creating a `my_example_dictionary` private dictionary via the API, you would make an API call by running this command:

```
curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'name=my_example_dictionary&write_only=true'
https://api.fastly.com/service/<service_id>/version/<version_number>/dictionary
```

which would return:

```
{
  "created_at": "2017-05-03T16:11:41+00:00",
  "deleted_at": null,
  "id": "<dictionary_id>",
  "name": "my_example_dictionary",
  "service_id": "<service_id>",
  "updated_at": "2017-05-03T16:11:41+00:00",
  "version": <version_number>,
  "write_only": true
}
```

After you've created the dictionary, you'll need to activate the service (/guides/edge-dictionaries/creating-and-using-dictionaries#activate-the-service-associated-with-the-dictionary-container) before you can start adding dictionary items (/guides/edge-dictionaries/creating-and-using-dictionaries#add-dictionary-items).

Viewing private dictionaries in VCL

The contents of private Edge Dictionaries are hidden in VCL. The private dictionary's metadata is displayed, as shown below:

```
table my_example_dictionary {
     # REDACTED dictionary content
     # last_updated: 2017-10-16 20:44:43
     # item_count: 2
     # digest: 8f92141234567890da30ba9cea7d98ef614
 }
```

• Guides (/guides/) > Developer's tools > Purging (/guides/purging/)

§ Authenticating URL purge requests via API (/guides/purging/authenticating-api-purge-requests)

Fastly's URL purge (/guides/purging/single-purges#purging-a-url) feature allows you to purge individual URLs on your website. By default, authentication is not required to purge a URL with the Fastly API, but you can enable API token (/guides/account-management-and-security/using-api-tokens) authentication in the Fastly web interface by adding a header or by using custom VCL.

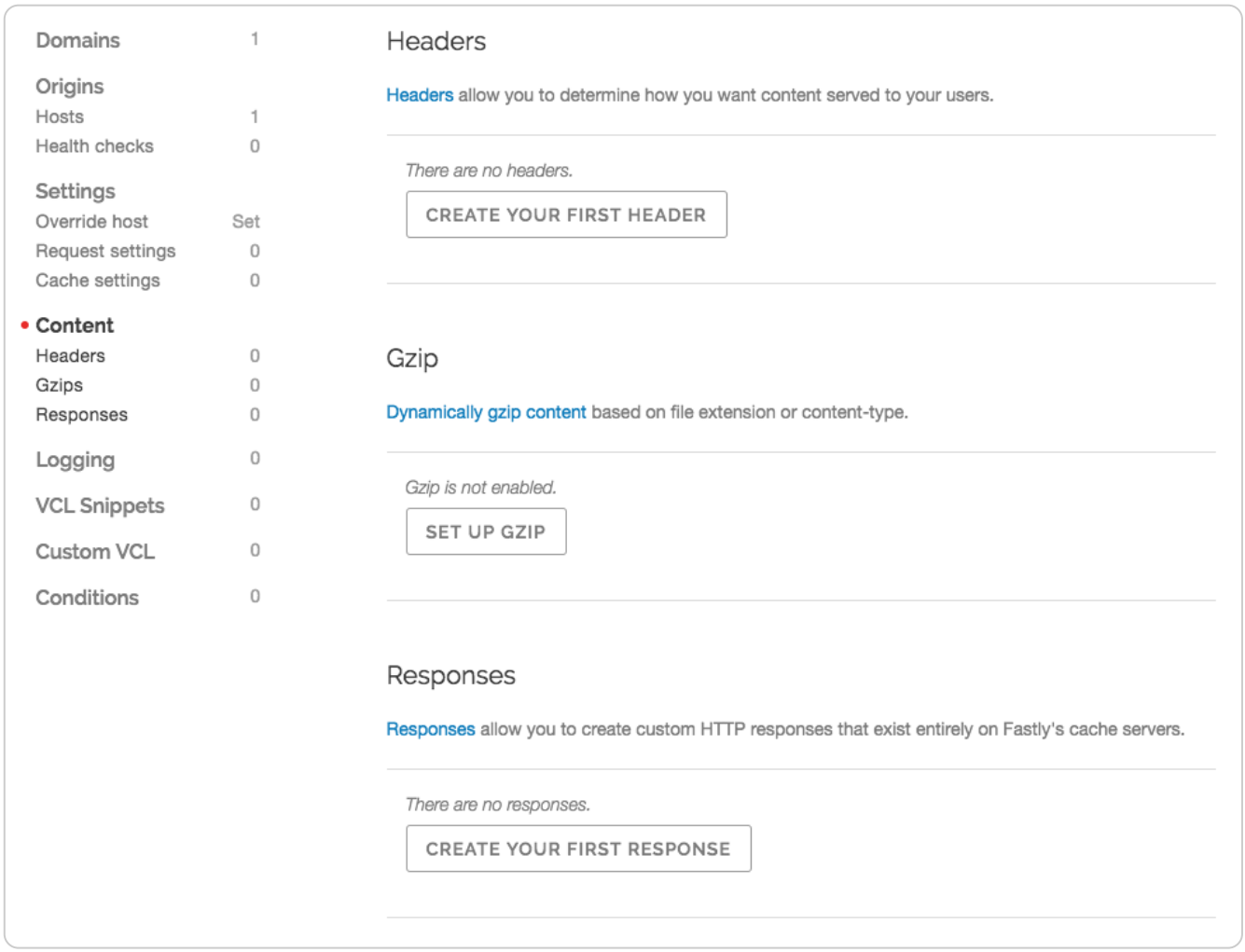
ℹ **NOTE:** All purge requests other than URL purges require authentication by default, as indicated in the API documentation (/api/purge#purge).

Enabling authentication in the Fastly web interface

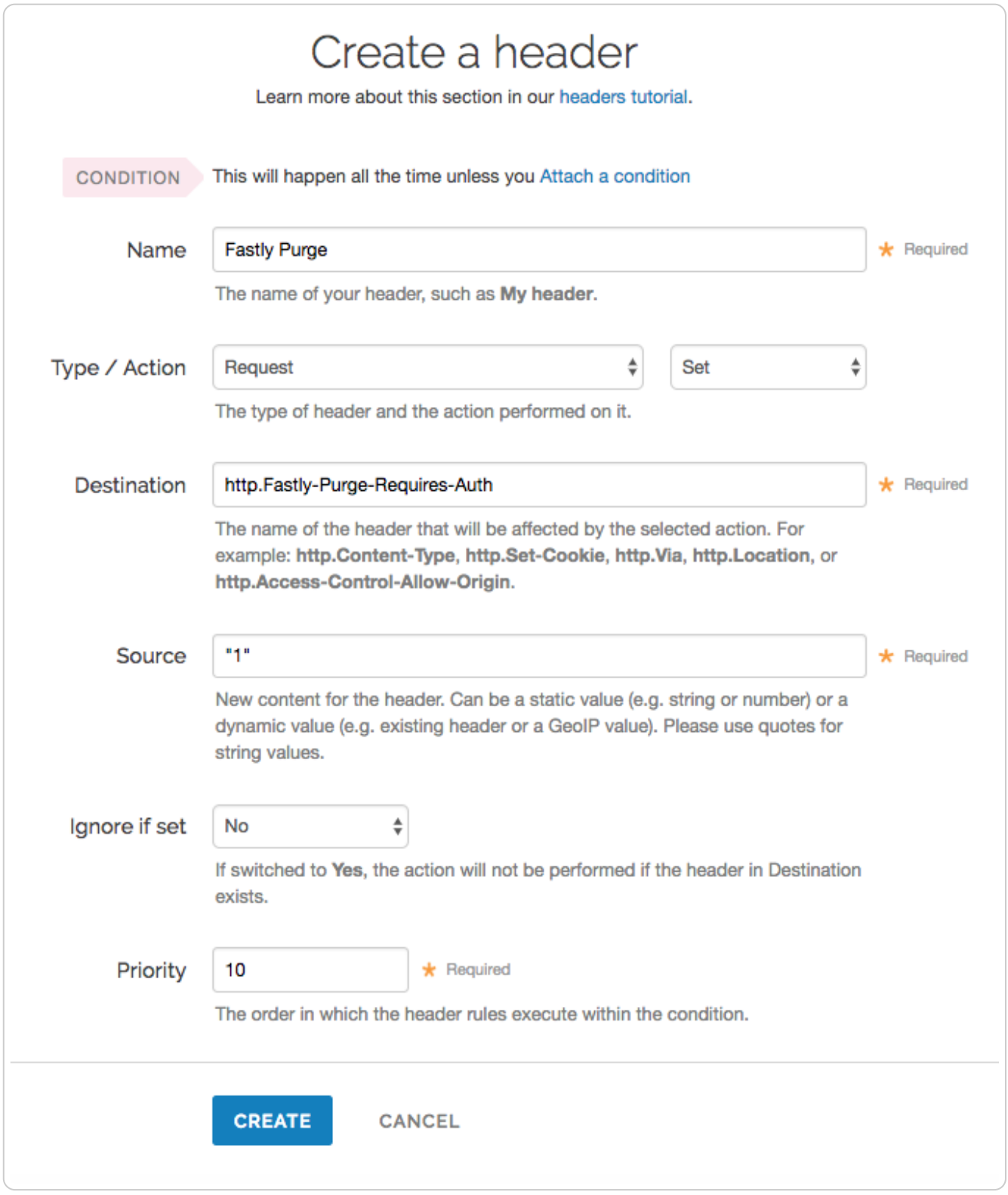
You can enable API token authentication for URL purge requests by adding a header and optionally attaching a condition in the Fastly web interface.

Adding the header

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `Fastly Purge`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Fastly-Purge-Requires-Auth`.
   - In the **Source** field, type `"1"`.

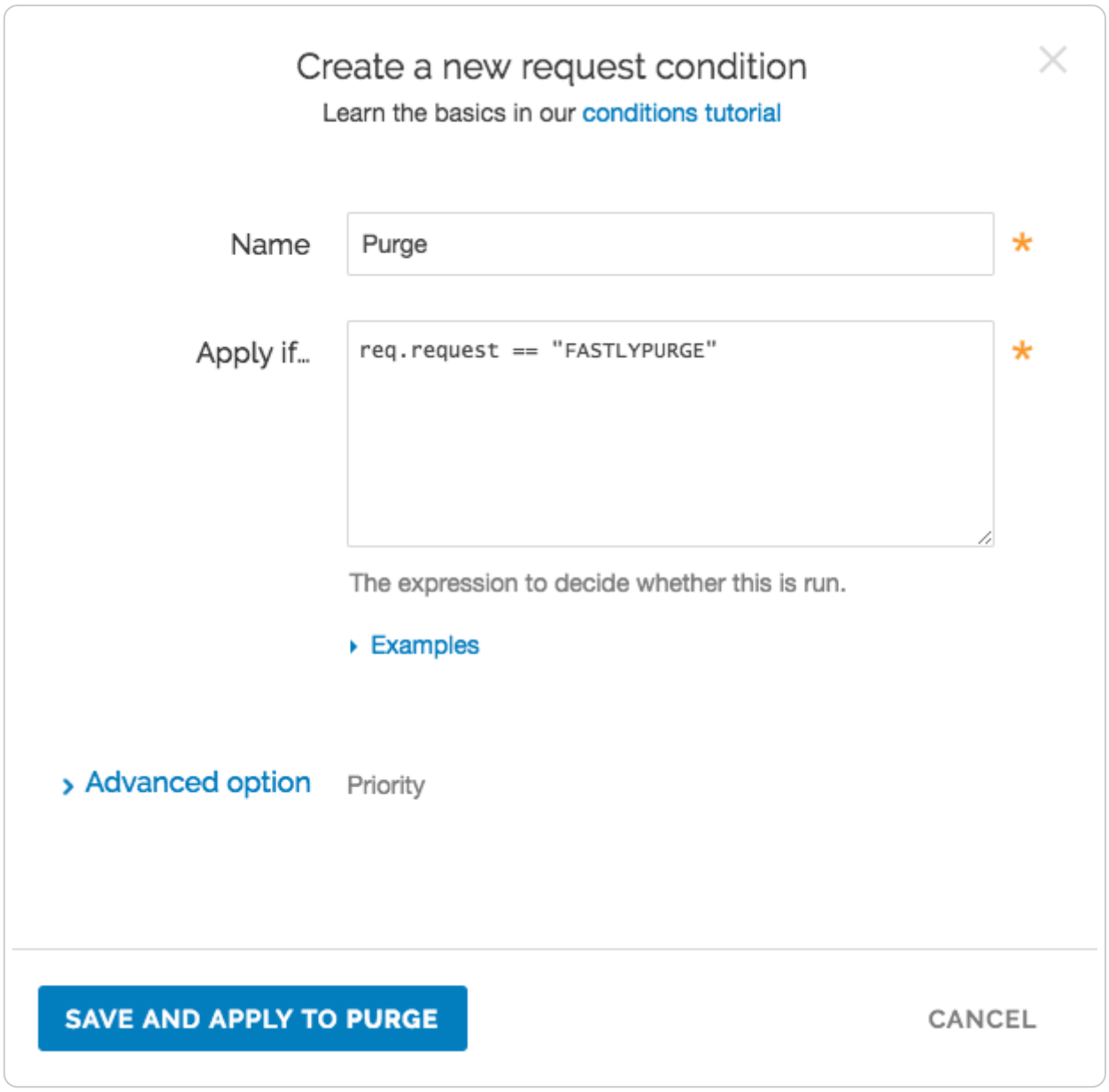
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button.

Attaching a condition

Attaching the following condition is optional. Without the condition, the header you just created will be added to all requests. With the condition, the header will be added to purge requests only.

1. On the Content page, click the **Attach a condition** link to the right of your new header. The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Purge`).
   - In the **Apply if** field, type `req.request == "FASTLYPURGE"`.

3. Click the **Save and apply to** button.
4. Click the **Activate** button to deploy your configuration changes.

Enabling authentication with custom VCL

If you'd rather enable API token authentication for URL purge requests using custom VCL (/guides/vcl/uploading-custom-vcl), add the following to your VCL file:

```
if (req.request == "FASTLYPURGE") {
  set req.http.Fastly-Purge-Requires-Auth = "1";
}
```

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Purging URLs with an API token

After you've enabled API token authentication for URL purge requests, you'll need to provide your API token (/guides/account-management-and-security/using-api-tokens) in the URL purge API request (/api/purge#purge_3aa1d66ee81dbfed0b03deed0fa16a9a):

```
curl -X PURGE -H Fastly-Key:FASTLY_API_TOKEN https://www.example.com/
```

which would return this response:

```
{"status": "ok", "id": "1234567890"}
```

WARNING: If your website is not configured to use HTTPS, we recommend purging using a POST request with a secure Fastly API URL. This will ensure that your API token in the header is encrypted in transit. The request will look like this: `curl -X POST -H Fastly-Key:FASTLY_API_TOKEN https://api.fastly.com/purge/<your_url_here>`.

---

§ Getting started with surrogate keys (/guides/purging/getting-started-with-surrogate-keys)

Efficient cache invalidation is an essential part of keeping your website fast. Purging too much cache using purge all (/guides/purging/single-purges#purging-all-content) may increase your website's load time while the cache rebuilds. If you find yourself purging all cache on more than a weekly basis, consider using surrogate keys for more targeted purging.

Surrogate keys allow you to selectively purge related content. Using the `Surrogate-Key` header, you can tag a group of objects with a key and then use it to purge multiple pieces of content at once. This process can occur automatically within your application, making it easier to cache and purge content that changes rapidly and unpredictably.

> ℹ **NOTE:** This guide assumes you're already familiar with the way content delivery networks (CDNs) work (/guides/basic-concepts/how-caching-and-cdns-work) in general, and the way Fastly's CDN works (/guides/basic-concepts/how-fastlys-cdn-service-works) in particular.

Understanding surrogate keys

After you've signed up for Fastly and added one or more services (/guides/basic-setup/working-with-services#creating-a-new-service), you can start examining how your origin server responds to requests. When your origin server responds to an HTTP request for content, it's because Fastly hasn't yet cached that content or the cache has expired. Your server's response to the request will resemble the example shown below. (Note that you can use the curl command (/guides/debugging/curl-and-other-caching-verification-methods) to inspect any of your server's responses.)

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: keep-alive
...
```

To control how your content is served to users and cached by Fastly, you can add to or modify the headers (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) that are included in your origin server's response. The `Surrogate-Key` header is one of the headers that you can add to the response. It allows you to "tag" an object, such as an image or a blog post, with one or more keys. When the object changes, you can reference the key in a purge request (/guides/purging/single-purges) to remove the object from the cache.

You can add space-delimited strings to the `Surrogate-Key` header, like this:

```
HTTP/1.1 200 OK
Surrogate-Key: key1 key2 key3
Content-Type: text/html
...
```

This response contains three surrogate keys: `key1`, `key2`, and `key3`. When Fastly receives a response like this, we use the surrogate keys to create a mapping from each key to the cached content, then we strip out the `Surrogate-Key` header so it's not included in the response to your readers.

Creating relationships between keys and objects

One of the major advantages of surrogate keys is that they allow for a many-to-many relationship between keys and objects. An origin server's response can associate multiple keys with the object, and the same key can be provided in different responses. Take a look at these two requests and responses:

```
GET /blog/ HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK Content-Type: text/html
Content-Length: 1234
Surrogate-Key: mainpage template-a
```

```
GET /blog/article/fastly-rocks HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2345
Surrogate-Key: template-a article-fastly-rocks
```

In this example, there are two objects (`/blog` and `/blog/article/fastly-rocks`) with three keys (`mainpage`, `template-a`, and `article-fastly-rocks`). Two of the keys (`mainpage` and `article-fastly-rocks`) are associated with a single object, and a third key (`template-a`) is associated with both objects.

Purging objects with surrogate keys

By using the `Surrogate-Key` header to associate keys with one or more objects, you can precisely control which objects are removed from cache during a purge. Consider the example presented above. Purging the `mainpage` key would remove only the `/blog` object from the cache. On the other hand, purging the `template-a` key would remove both the `/blog` and `/blog/article/fastly-rocks` objects from the cache.

You can use the Fastly web interface to manually purge objects via key (/guides/purging/single-purges#purging-with-keys), or you can use our Purge API (/api/purge) to purge a collection of objects with one API call. If you're using Fastly to cache your API, check out the guide on purging

API cache with surrogate keys (/guides/api-caching/purging-api-cache-with-surrogate-keys) to learn how surrogate keys can help you purge API cache.

Looking at a practical example

Let's look at a practical example to learn how surrogate keys work. Imagine you're building a picture-hosting website and you're using Fastly to speed it up. You initially decide to cache picture pages by their URL (e.g., `http://www.example.com/pic/id`). When a picture's information changes, you'll remove the old version by sending a purge request to the Fastly API:

```
PURGE /pic/{id}
Host: example.com
Accept: */*
```

But there's a potential problem with this solution. You display a user's information next to their pictures, so you'll need to purge all of the user's picture pages if they change their information. You could send an individual purge for each picture, but that would take too long. As an alternative, you could cache the pages for a very short amount of time, but that would waste our server resources.

Surrogate keys solve this problem. By adding a surrogate key to all of a user's picture pages (e.g., `/user/542`, `/user/25`), you can purge all of the user's pictures by sending Fastly a purge for the user's surrogate key when they update their information. Now, instead of having to purge each picture individually, you can update them all with just one request:

```
PURGE /service/id/purge/user/542
...
```

Purging multiple sites at the same time

What if you wanted to build a mobile version of the picture-hosting website to complement the desktop version? You'll need a way to purge both the desktop version and the mobile version at the same time. Surrogate keys can help in this instance. You can tag the different versions of a picture page with the same surrogate key (e.g., `pic/76`, `pic/345`) and purge them all at once. All of the related content on our sites can now be purged with one request.

Tagging templates with surrogate keys

Surrogate keys come in really handy when making changes to templates. Imagine you have to make a change to the banner of the website. Since you're caching entire page, updating the header template isn't enough. You'll also need to purge all the pages that use the template. You could purge every page on the website, but there's no reason to purge content that doesn't use the header template.

You can make things easy by using surrogate keys. By adding surrogate keys for each template on a page (e.g., `/templates/pic/show`, `/templates/pic/header`, `/templates/pic/comment`), you can check which templates have changed and purge only pages with modified templates.

Generating and setting surrogate keys

There are two ways to set the `Surrogate-Key` header: by adding the header in the Fastly web interface, or by generating the keys with your own application. We describe how to use the Fastly web interface in our guide to generating Surrogate-Key headers based on URLs (/guides/purging/setting-surrogate-key-headers-based-on-a-url) (we have a separate guide for Amazon S3 origins (/guides/purging/setting-surrogate-key-headers-for-amazon-s3-origins)).

Automatically generating keys with your own application is described below using Fastly's Test Blog application (https://github.com/fastly/fastly-test-blog) as an example. The test blog is a Ruby on Rails application that comes preloaded with example content that can be cached and purged using Fastly via the fastly-rails (https://github.com/fastly/fastly-rails) Ruby gem. (We also have other API clients (/api/clients) that support surrogate keys.)

> ℹ **NOTE:** You can install the Fastly Test Blog and recreate this example yourself to practice generating surrogate keys. For the purposes of this example, we've made one deviation from the Fastly Test Blog instructions outlined on the GitHub page (https://github.com/fastly/fastly-test-blog). We've added a CNAME record (/guides/basic-setup/adding-cname-records) for our test blog to create two URLs — http://origin.example.com, which is the URL of the origin server not attached to our Fastly service, and http://fastly.example.com, which is the URL being cached by Fastly. Having the two URLs available will be useful when we examine the headers.

Configuring the API client

The fastly-rails (https://github.com/fastly/fastly-rails) gem provides a `set_surrogate_key_header` method (https://github.com/fastly/fastly-rails#headers) which the test blog uses to automatically generate surrogate keys for new articles. You can see how this works by examining the code in articles_controller.rb (https://github.com/fastly/fastly-test-blog/blob/master/app/controllers/articles_controller.rb). A slightly modified excerpt of the code from the controller is shown below.

```
class ArticlesController < ApplicationController
  # include this before_action in controller endpoints that you wish to edge cache
  # This can be used with any custom actions. Set these headers for GETs that you want to cache
  # e.g. before_action :set_cache_control_headers, only: [:index, :show, :my_custom_action]
  before_action :set_cache_control_headers, only: [:index, :show]

  # Returns all Article objects, and sets a table_key of 'articles',
  # and a record_key for each article object: "#{table_key}/#{article_id}"
  def index
    @articles = Article.all
    set_surrogate_key_header 'articles', @articles.map(&:record_key)
  end

  # Sets a surrogate key for the current article.
  #
  # Example:
  #
  #   Article[75]
  #   Surrogate-Key:articles/75
  def show
    set_surrogate_key_header @article.record_key
  end

  ...
end
```

The `before_action` method creates `Cache-Control` and `Surrogate-Control` HTTP headers with a default TTL of 30 days. This method must be added to any controller action that you want to edge cache. The `set_surrogate_key_header` method sets `Surrogate-Key` headers for objects that you want to be able to purge. In this case, surrogate keys are set for each article and the articles index.

Examining the headers

Now that you've looked at how the surrogate keys are generated by the test blog, inspect the headers on an article page on the origin server. Here's the partial output from `curl -svo /dev/null origin.example.com/articles/1`:

```
HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Cache-Control: public, no-cache
Surrogate-Control: max-age=86400
Surrogate-Key: articles/1
```

Notice how the `Cache-Control`, `Surrogate-Control`, and `Surrogate-Key` headers are present in the response shown above. Thanks to the `set_surrogate_key_header` method, the test blog application automatically generates the unique `articles/1` surrogate key for this article.

Next, inspect the headers on the article index page on the origin server URL. Because this page lists all of the articles, you might expect it to contain the surrogate keys for every article displayed on the page, and that is indeed the case. Here's the partial output from `curl -svo /dev/null origin.example.com`:

```
HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Cache-Control: public, no-cache
Surrogate-Control: max-age=86400
Surrogate-Key: articles articles/1 articles/2 articles/3 articles/4 articles/5 articles/6 artic
les/7 articles/8 articles/9 articles/10 articles/11 articles/12 articles/15 articles/16 article
s/17 articles/18 articles/19 articles/20 articles/27 articles/28 articles/29 articles/30 articl
es/31
```

If you purged the `articles/1` surrogate key, both http://origin.example.com/articles/1 and http://origin.example.com would be purged from the cache.

Finally, take a look at the URL that's piped through Fastly: http://fastly.example.com. The surrogate keys won't be visible in the headers, but Fastly knows what they are. Recall from understanding surrogate keys that Fastly strips out the `Surrogate-Key` header and creates a mapping from each key to the cached content. Here's the partial output from `curl -svo /dev/null fastly.example.com`:

```
HTTP/1.1 200 OK
X-Frame-Options: SAMEORIGIN
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Cache-Control: public, no-cache
Content-Type: text/html; charset=utf-8
Via: 1.1 varnish
Age: 78309
X-Served-By: cache-iad2120-IAD
X-Cache: HIT
X-Cache-Hits: 1
X-Timer: S1449255701.272992,VS0,VE5
```

With the `Surrogate-Key` header present on the index and article pages, you're now able to manually purge blog pages via key (/guides/purging/single-purges#purging-with-keys), or purge a collection of pages with one API call (/api/purge).

Limitations

The surrogate keys sent by your origin server can be as simple or complex as you need, but there are a couple limitations:

- Surrogate keys are limited to a total length of 1,024 bytes each
- Any keys that exceed the limit will be dropped instead of truncated
- Surrogate key headers are limited to a total length of 16,384 bytes
- Any keys past the one that exceeds the limit will be dropped

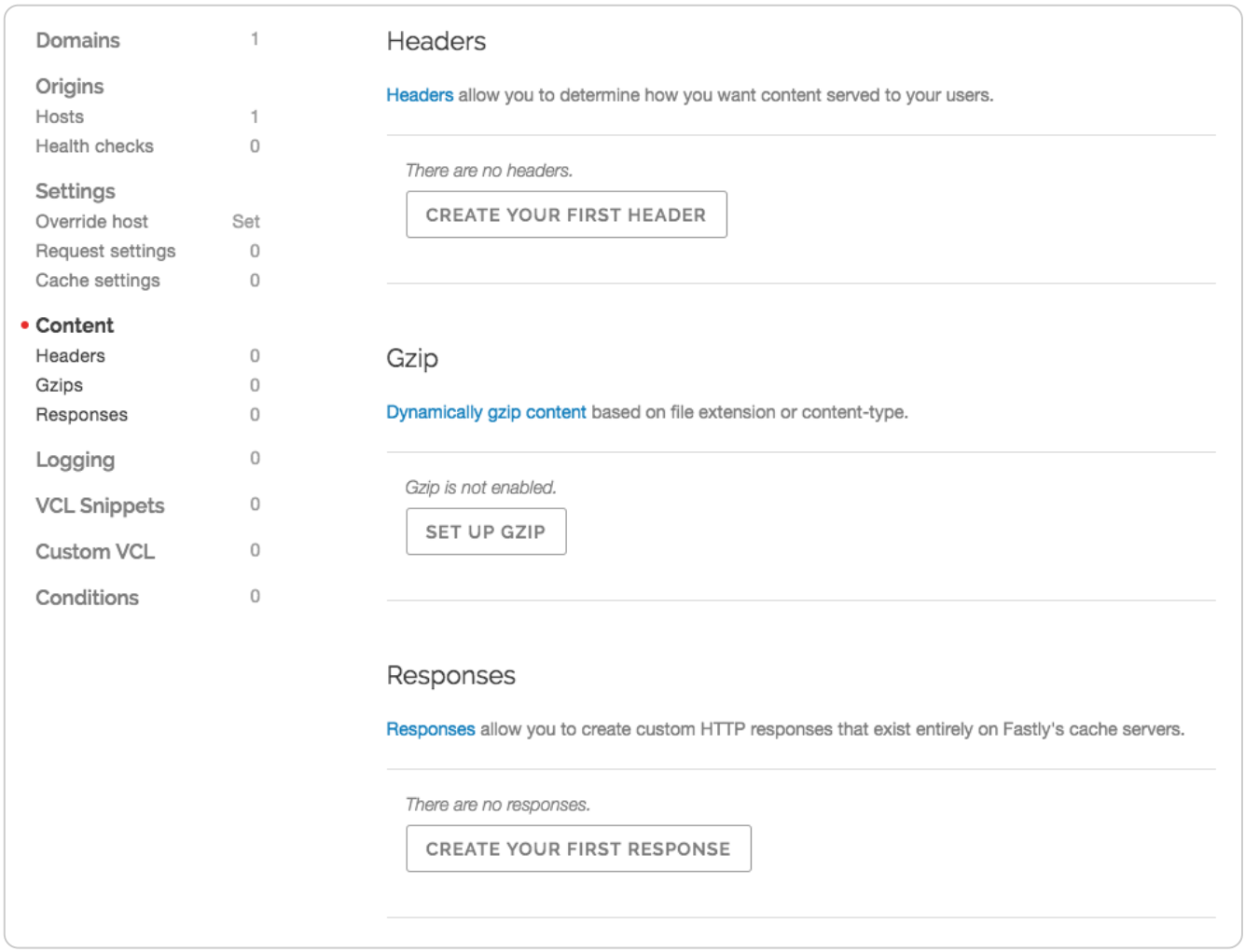
---

§ Setting Surrogate-Key headers based on a URL (/guides/purging/setting-surrogate-key-headers-based-on-a-url)

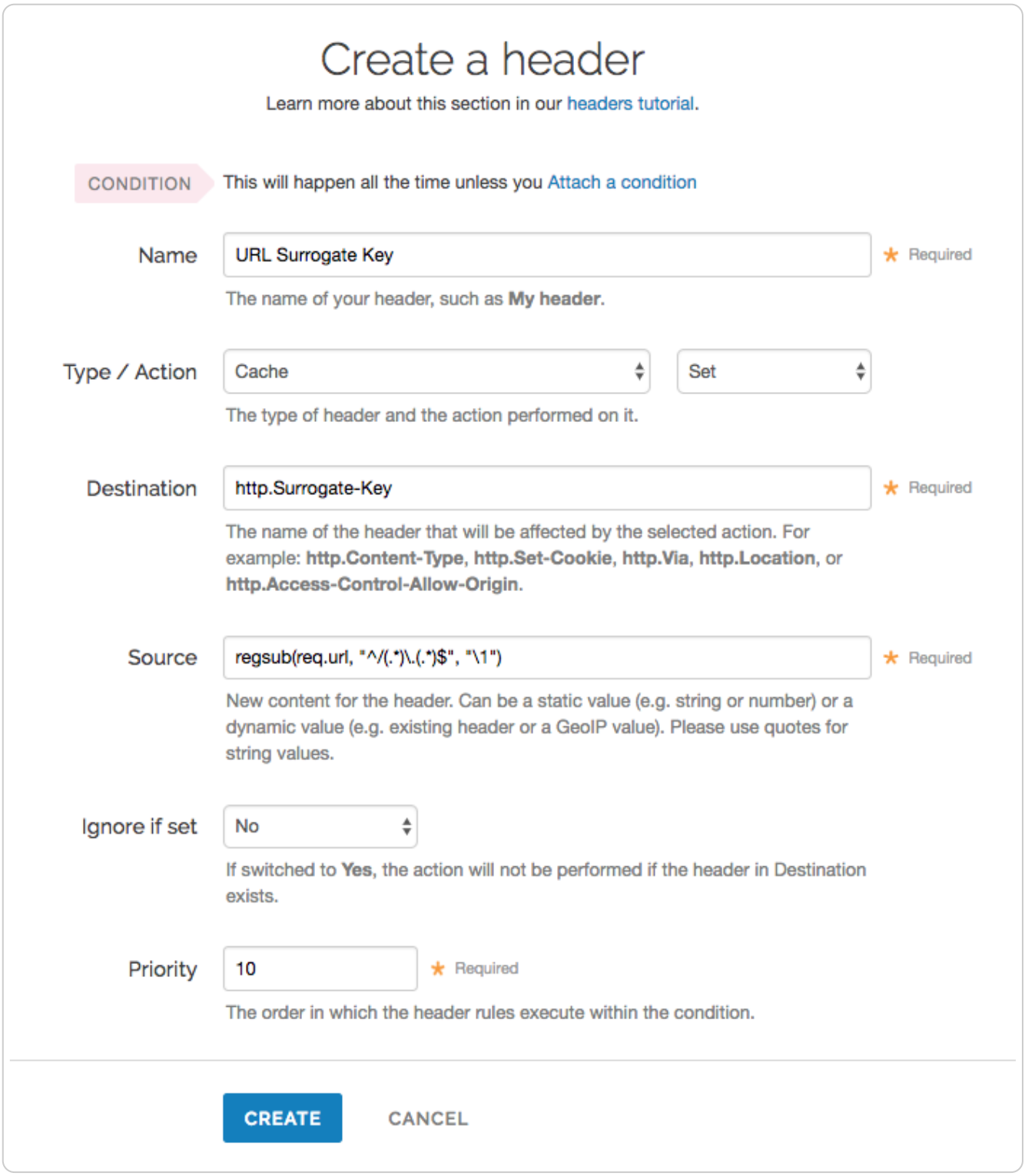
You can mark content with a surrogate key (/guides/purging/getting-started-with-surrogate-keys) and use it to purge groups of specific URLs (/guides/purging/single-purges#purging-with-keys) at once without purging everything (/guides/purging/single-purges#purging-all-content), or purging each URL (/guides/purging/single-purges#purging-a-url) singularly.

Follow these instructions to set Surrogate-Key headers based on a URL:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a human-readable name for the header. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Surrogate-Key`.

- In the **Source** field, type `regsub(req.url, "^/(.*)\.(.*)$", "\1")`. This will accept a URL that looks like `/foo.html` and will create the Surrogate-Key `foo`.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button to create your header.
8. Click the **Activate** button to deploy your configuration changes.

ℹ **NOTE:** There are several limitations to surrogate keys. See the surrogate key limitations (/guides/purging/getting-started-with-surrogate-keys#limitations) section for more information.

---

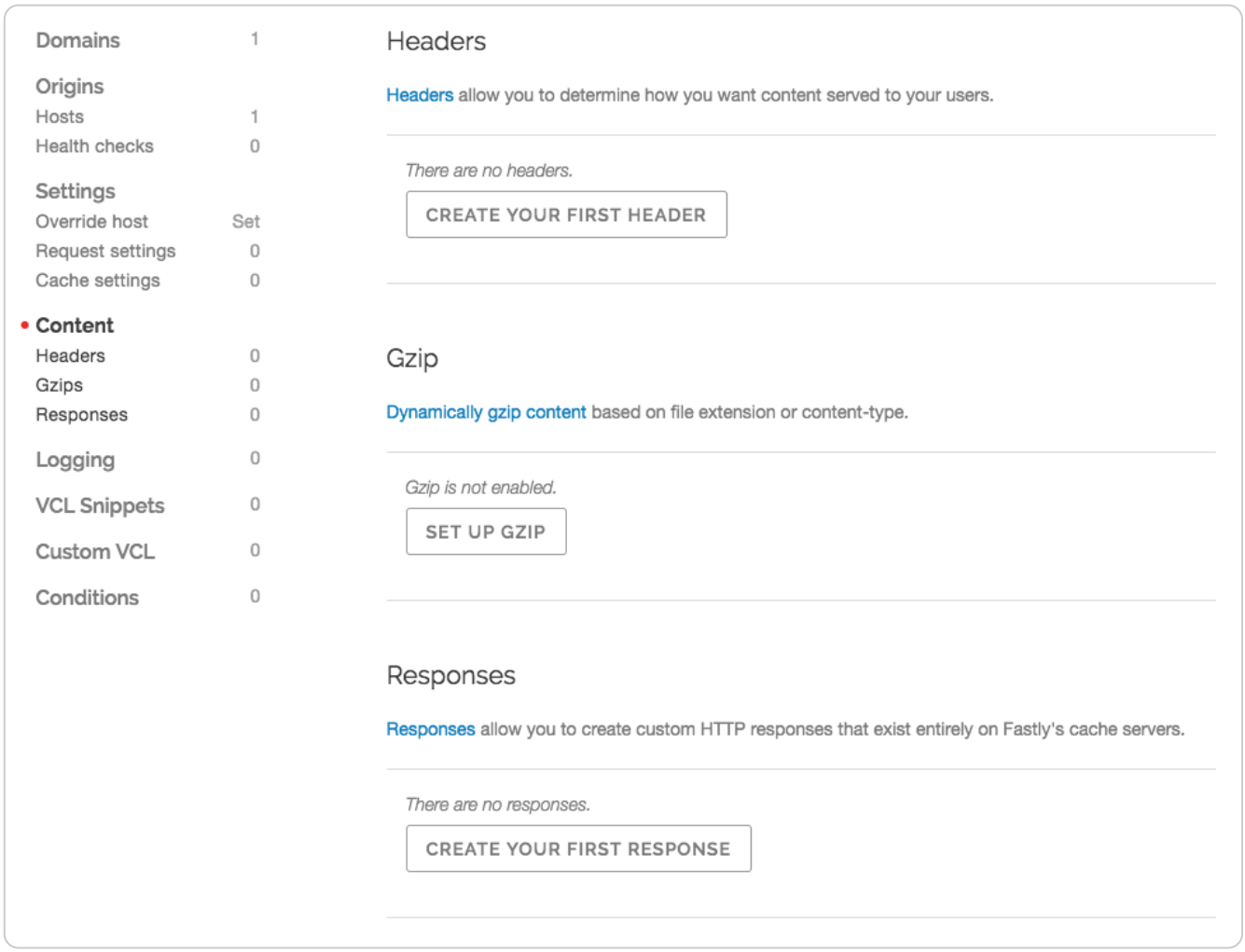
§ Setting Surrogate-Key headers for Amazon S3 origins (/guides/purging/setting-surrogate-key-headers-for-amazon-s3-origins)

You can mark content with a surrogate key (/guides/purging/getting-started-with-surrogate-keys) and use it to purge groups of specific URLs (/guides/purging/single-purges#purging-with-keys) at once without purging everything (/guides/purging/single-purges#purging-all-content), or purging each URL (/guides/purging/single-purges#purging-a-url) singularly. On the Amazon S3 side, you can use the `x-amz-meta-surrogate-key` header to mark your content as you see fit, and then on the Fastly side set up a Header configuration to translate the S3 information into the header we look for.

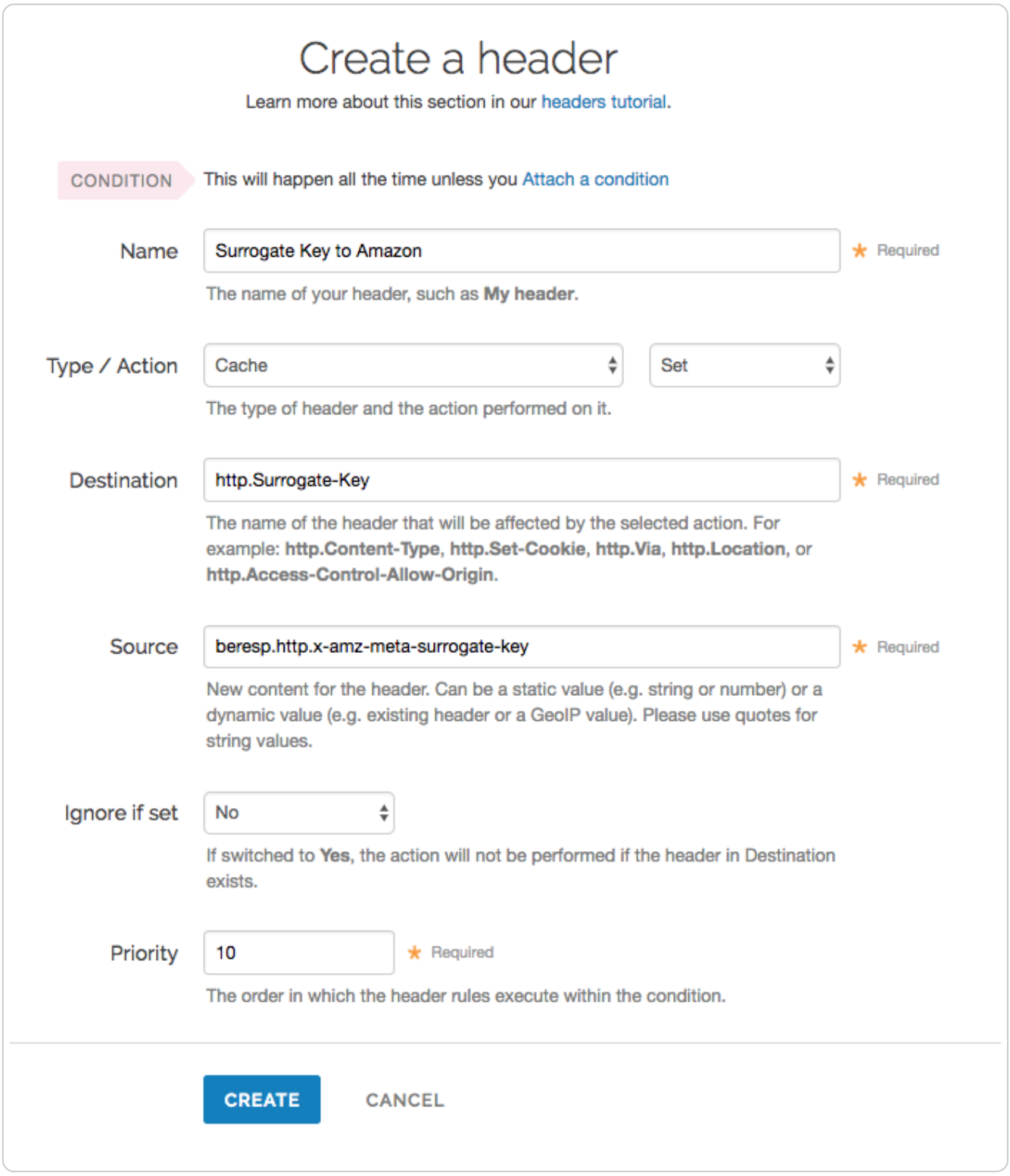
❗ **IMPORTANT:** Pay close attention to the capitalization. Amazon S3 only accepts all lowercase header names.

Follow these instructions to set Surrogate-Key headers for Amazon S3 origin servers:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a human-readable name for the header. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Surrogate-Key`.
   - In the **Source** field, type `beresp.http.x-amz-meta-surrogate-key`.

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button to create your header.
8. Click the **Activate** button to deploy your configuration changes.

NOTE: There are several limitations to surrogate keys. See the surrogate key limitations (/guides/purging/getting-started-with-surrogate-keys#limitations) section for more information.

---

§ Single purges (/guides/purging/single-purges)

Fastly provides several levels of cache purging. You can purge something as small as a single URL via the "Purge URL" command or as large as all content under a service via the "Purge All" command. You can also selectively purge content via key-based purging using the "Purge Key" command. We also provide a purging feature called Soft Purge that allows you to mark content as outdated (stale) instead of permanently deleting it from Fastly's caches.

TIP: To mark content as outdated instead of permanently deleting it, check out our Soft Purge (/guides/purging/soft-purges) feature. You may also be interested in our wildcard purging (/guides/purging/wildcard-purges).

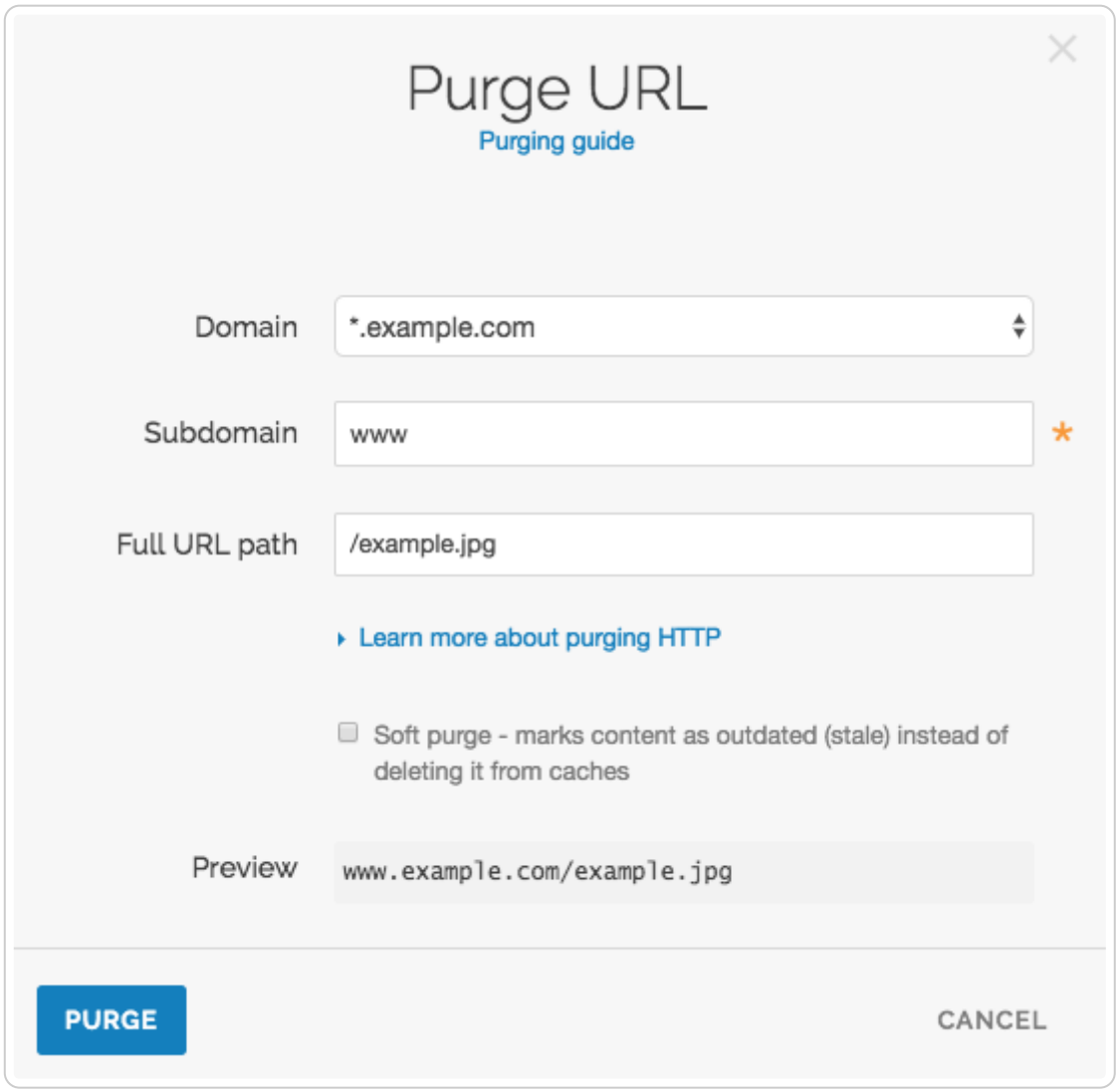
Purging via the user interface

To purge content using the Fastly web interface, choose one of the purging methods below.

Purging a URL

To purge a single URL, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. From the **Purge** menu, select **Purge URL**. The Purge URL window appears.

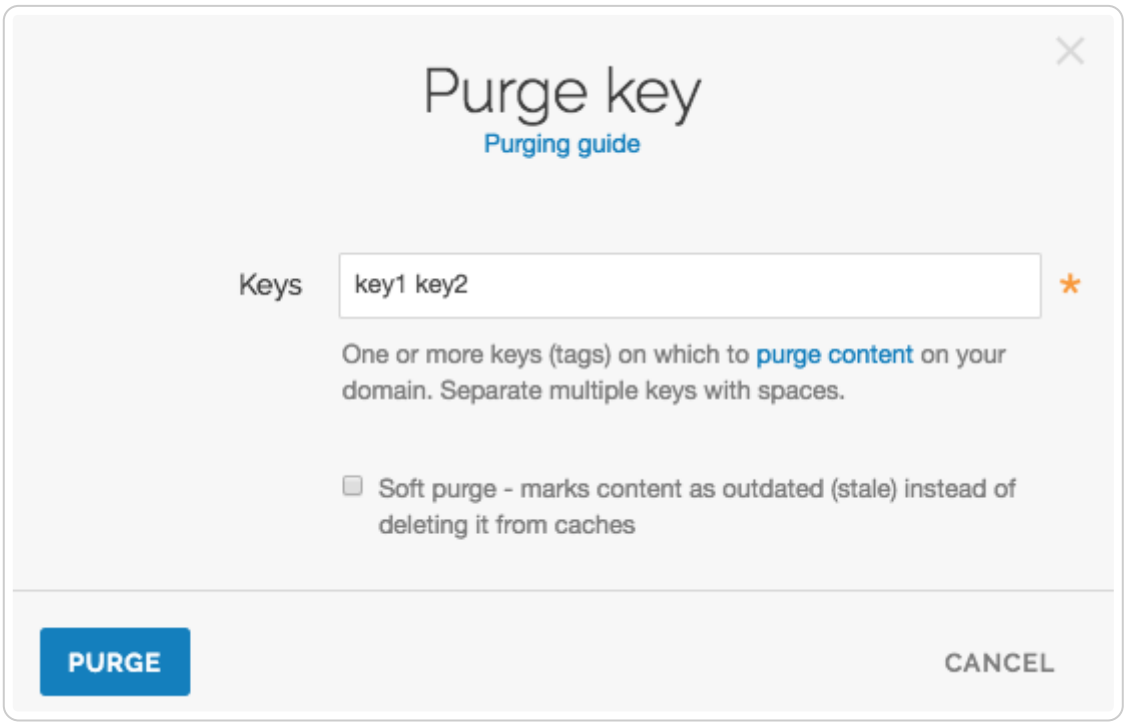


4. From the **Domain** menu, select the domain (/guides/basic-configuration/working-with-domains) on which your content resides. If the domain you select is a wildcard domain (e.g., `*.example.com`) the Subdomain field will appear.
5. If the **Subdomain** field appears, type the subdomain to purge for the wildcard domain you've selected (e.g., `www`).
6. In the **Full URL path** field, type the path to the content you'll be purging (e.g., `/example.jpg`). The Preview field displays the URL that will be purged.
7. Optionally select the Soft purge (/guides/purging/soft-purges) checkbox to mark your content as outdated instead of deleting it from cache.
8. Click the **Purge** button.

Purging with keys

To purge content with surrogate keys (/guides/purging/getting-started-with-surrogate-keys), follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. From the **Purge** menu, select **Purge Key**. The Purge Key window appears.



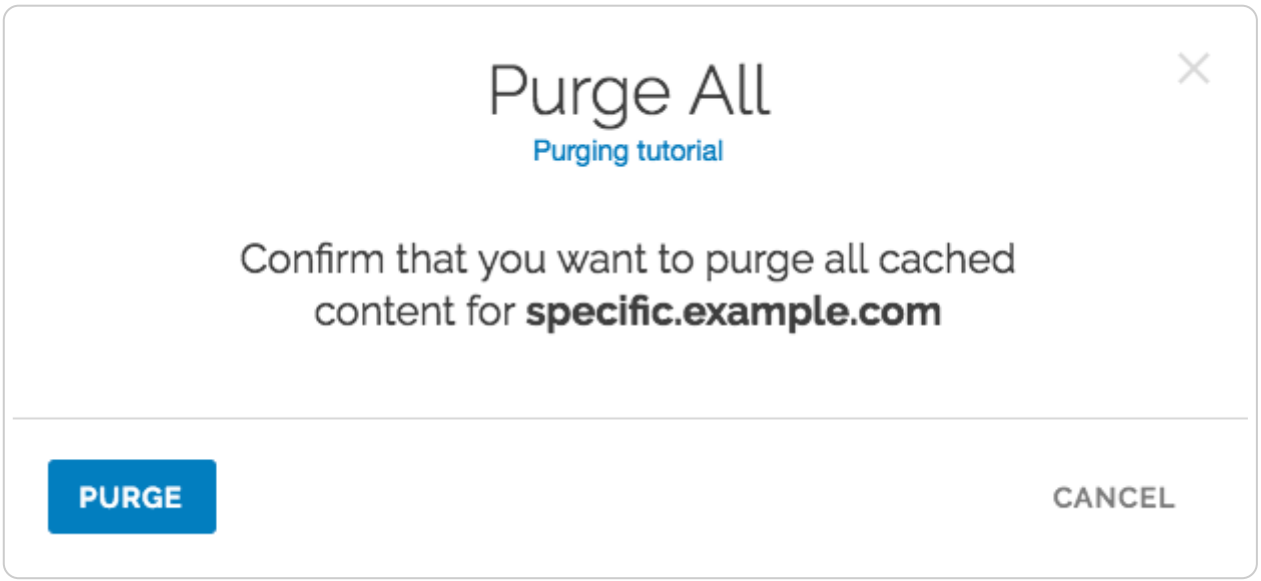
4. In the **Key** field, type one or more surrogate keys. Use spaces to separate multiple keys.
5. Optionally select the Soft purge (/guides/purging/soft-purges) checkbox to mark your content as outdated instead of deleting it from cache.
6. Click the **Purge** button.

Purging all content

⚠ **WARNING:** Do not purge all cached content if you are seeing 503 errors (/guides/debugging/common-503-errors). Purge all overrides stale-if-error (/guides/performance-tuning/serving-stale-content) and increases the requests to your origin server, which could result in additional 503 errors.

To instantly purge all content under your service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. From the **Purge** menu, select **Purge All**. The Purge All window appears and displays the exact domain you'll be purging.



4. Click the **Purge** button.

Purging via API

The syntax for purging a service through the API can be found in the Purging section (/api/purge) of the API (/api/) documentation.

---

§ Soft purges (/guides/purging/soft-purges)

Fastly provides a Soft Purge feature that allows you to mark content as outdated (stale) instead of permanently purging and thereby deleting it from Fastly's caches. Objects invalidated with Soft Purge will be treated as outdated (stale) while Fastly fetches a new version from origin. You can purge by URL or by surrogate key using Soft Purge (/guides/purging/single-purges).

Before using Soft Purge, we recommend you implement one of the following revalidation methods:

- Set up `ETag` or `Last-Modified` headers for relevant content on your origin servers.
- Configure `stale_while_revalidate` to serve stale content (/guides/performance-tuning/serving-stale-content) and fetch the newest version of the object from origin in the background. If you choose this revalidation method, you must also configure `stale_if_error` at the same time.

To implement Soft Purge, add a `Fastly-Soft-Purge` request header (such as `Fastly-Soft-Purge: 1`) to any single URL or key-based purge. For example, to purge the URL `www.example.com` with Soft Purge, you would issue the following command:

```
curl -X PURGE -H "Fastly-Soft-Purge:1" http://www.example.com
```

§ Wildcard purges (/guides/purging/wildcard-purges)

Wildcard purging allows you to flush the cache of all pages under a directory branch or URL path; for example, you want to empty the cache of all pages under your "/service" path. Having to purge each URL (/guides/purging/single-purges#purging-a-url) one by one using the Fastly API (/api/purge) or via the Fastly app is not very efficient.

Although Fastly does not have a specific wildcard purge function, you can implement the same behavior by making a small configuration change using surrogate keys (/guides/purging/getting-started-with-surrogate-keys). Surrogate keys allow you to tag a group of objects with a keyword (key) and then purge multiple pieces of content at once with it via the web interface or via custom VCL.

> ❗ **IMPORTANT:** Purging will only apply to new objects as they're being put into the cache after you set up configuration changes. It will not apply to objects already in the cache when this configuration is being applied.

To purge content based on wildcard paths, follow the steps below.

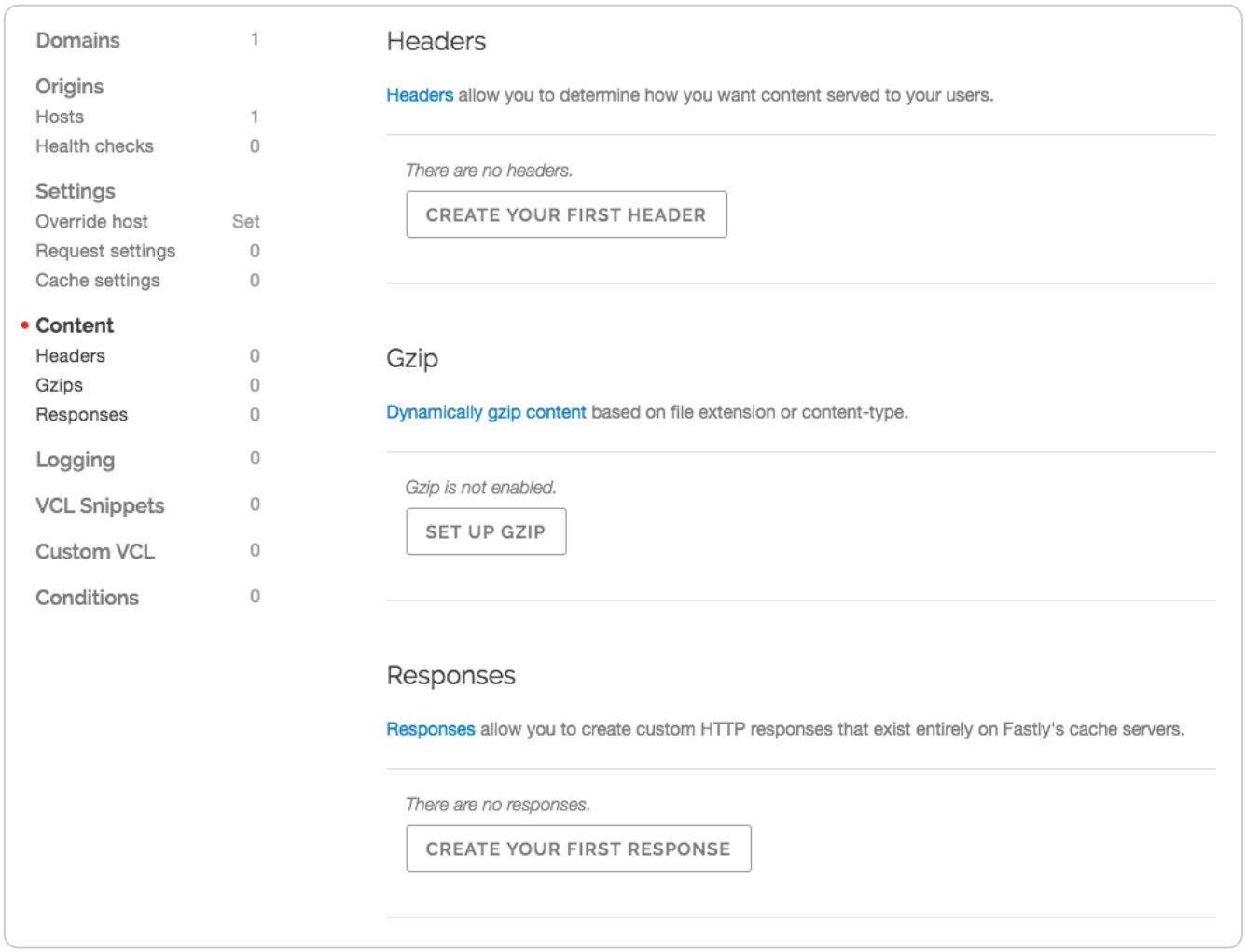
Via the web interface

To purge content based on wildcard paths via the web interface, follow the steps below.

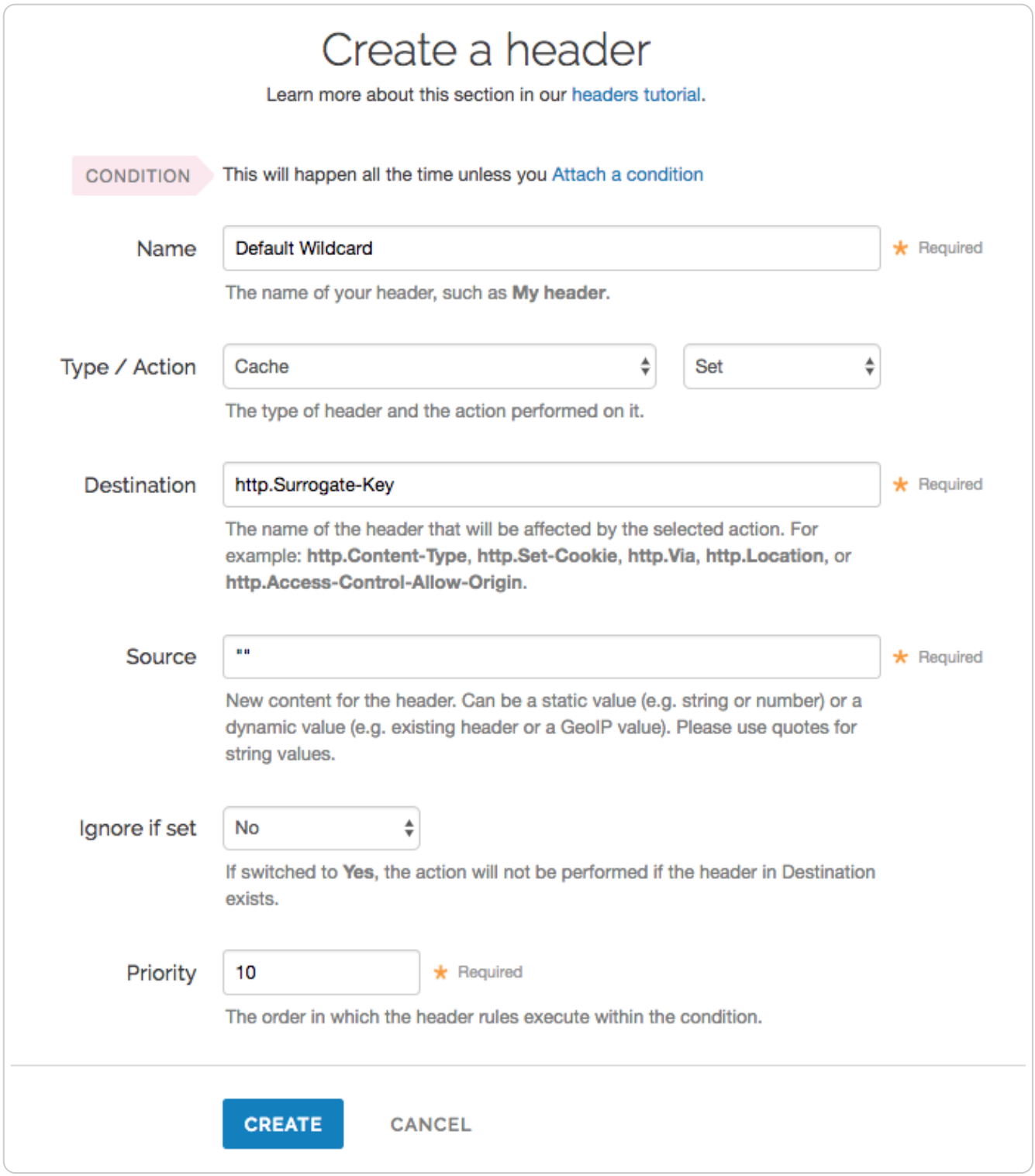
Create a default wildcard header

We set a default wildcard so that we have the flexibility to append other surrogate keys to a URL path.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Default Wildcard`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache** and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Surrogate-Key`.
   - In the **Source** field, type `""`.

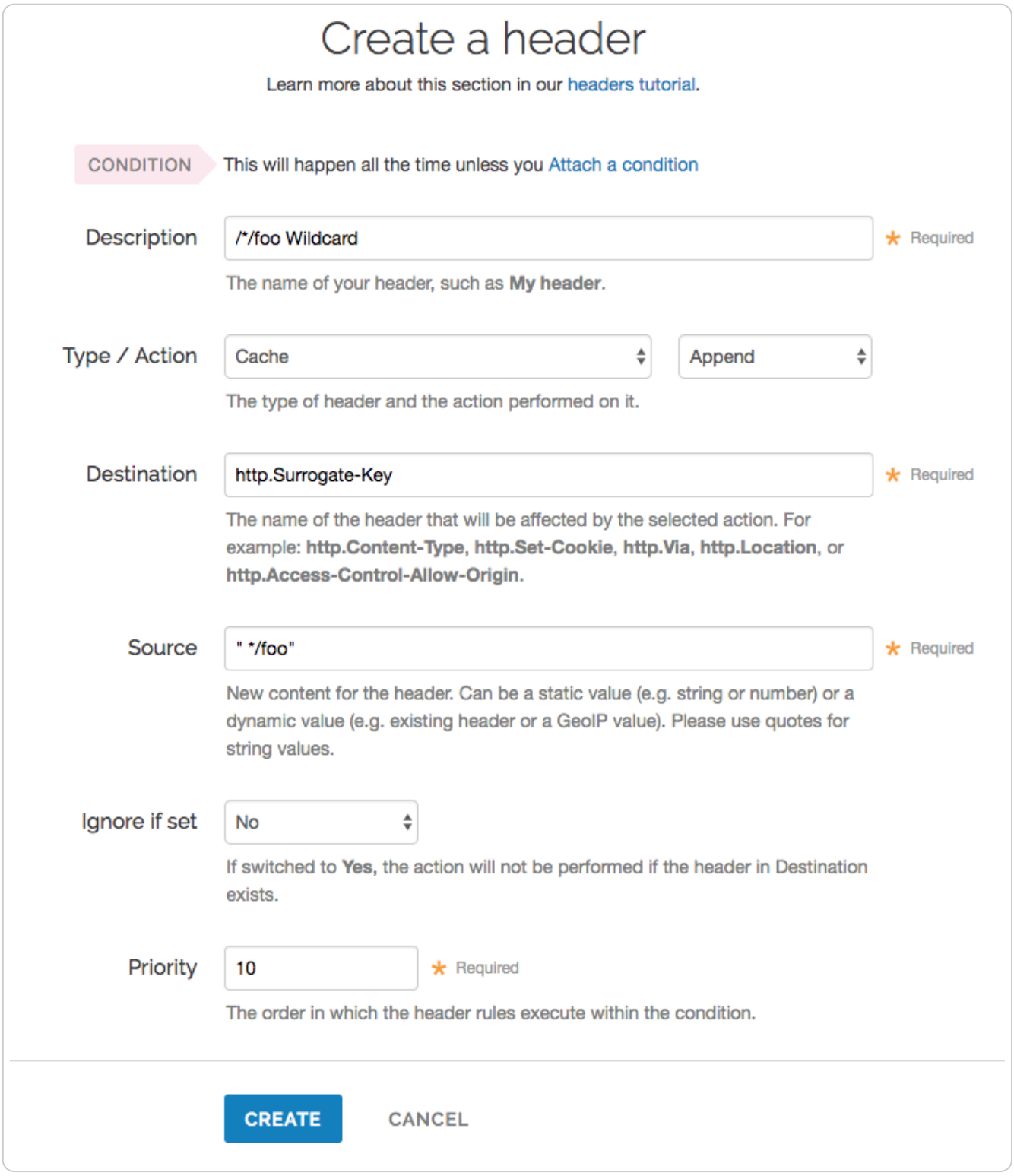
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button. A new header appears in the Headers area of the Content page.

Create headers for each wildcard path being purged

Next, create a header for each of the wildcard paths you need the ability to purge. For instance, you want to purge the wildcard path `/*/foo`.

1. Click the **Create header** button to create another new header.



2. Fill out the **Create a header** fields as follows:
   - In the **Description** field, type `/*/foo Wildcard`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Append**.
   - In the **Destination** field, type `http.Surrogate-Key`.

- In the **Source** field, type `" */foo"`. There is a space before the asterisk in the Source field, which is important when appending multiple surrogate keys to a URL.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `20`.

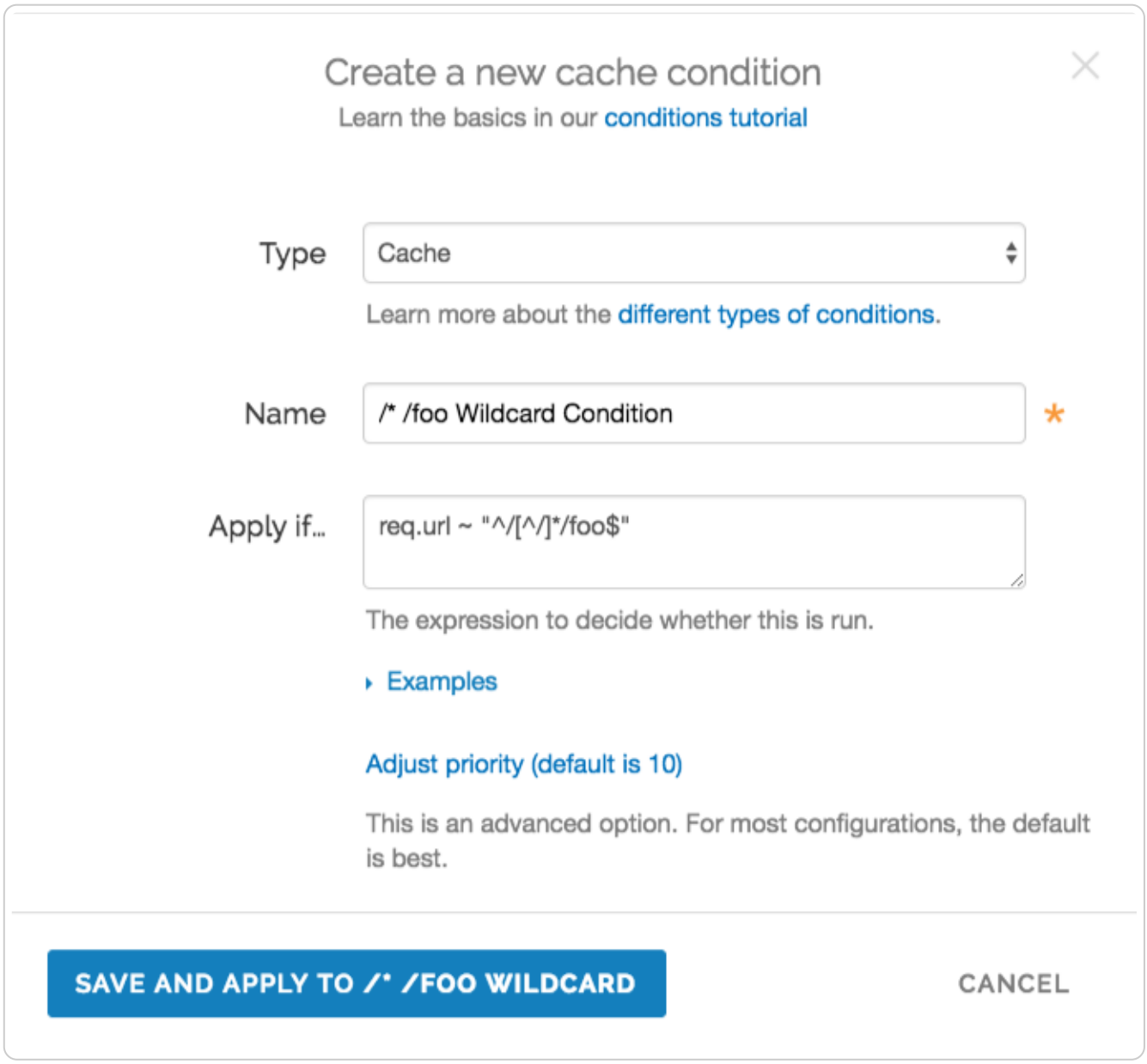
3. Click the **Create** button. A new header appears in the Headers area of the Content page.

Notice the Action is set to Append to add to the default wildcard surrogate key. The Priority is set to 20 so that the Default Wildcard header is executed first and then the wildcard path appends.

Create conditions for each wildcard path being purged

Finally, create a condition for each of the wildcard paths you need the ability to purge.

1. Click the **Attach a condition** link next to the wildcard path header name. The Create a new cache condition window appears.



2. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, type `/*/foo Wildcard Condition`.
   - In the **Apply if** field, type `req.url ~ "^/[^/]*/foo$"`.
3. Click the **Save and apply to** button to create the new condition.

What does the condition mean? In the Apply if field above, the first `"^"` and `"$"` tells Fastly to look for the following pattern:

- Start from the first slash after the request host header.
- There should be one directory.
- It should be followed by the path `/foo` ending the URL.

Some examples would be `/a/foo`, `/bar/foo`, and `/c/foo`. You could also remove the first `"^"` and `">"$"` to allow the condition to be more general so that the pattern can occur in the middle of a URL path.

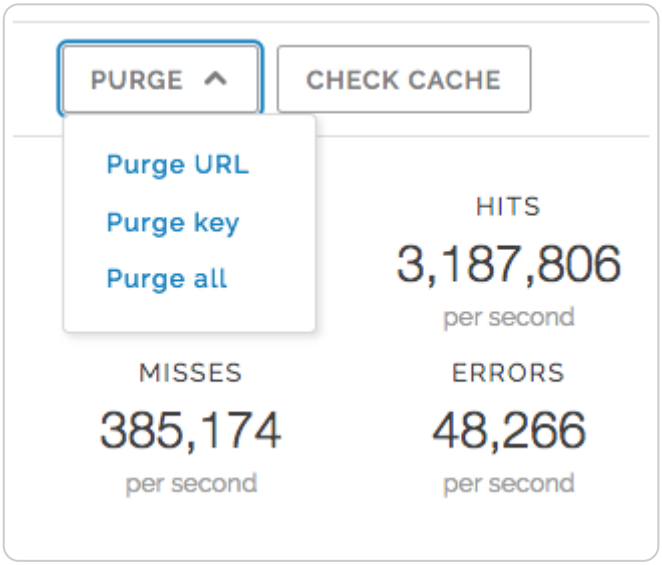
Some other examples for URL wildcard conditions:

Apply if field	Matched pattern
`req.url ~ "/[^/]*/foo"`	`/delta/wow/a/foo/neat/cool/img.gif`
`req.url ~ "^/.*/foo$"`	`/a/b/c/d/e/f/foo`

Purge the wildcard

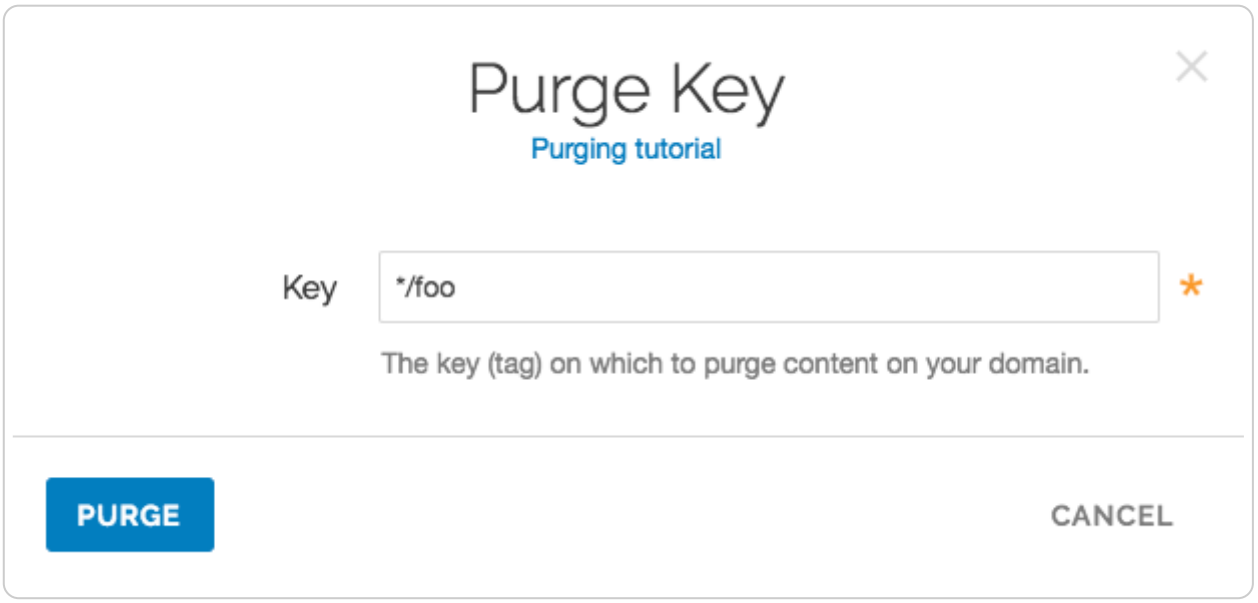
Ready to purge that wildcard? You can do this through the UI using the steps below.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the **Purge** menu, select **Purge Key**.



   The Purge Key window appears.

3. In the **Keys** field, type the surrogate key you want to purge. Continuing with our example, you would type `*/foo` without the quotes that were entered in the Source field of the New Header window above.



4. Click the **Purge** button.

Via custom VCL

To purge content based on wildcard paths via custom VCL, follow the steps below.

1. Add the following code to the VCL template:

```
sub construct_skey {
  if (req.url.path ~ "^((((((/[^/]+)?/[^/]+)?/[^/]+)?/[^/]+)?/[^/]+)?/[^/]+)") {
    # This prevents us from doing this twice when shielding
    if (std.strstr(beresp.http.Surrogate-Key, re.group.1)) {
      return;
    }

    if (!re.group.2) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-
Key " ", "")
        + re.group.1;
      return;
    }

    if (!re.group.3) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-
Key " ", "")
        + re.group.1 + " " + re.group.2;
      return;
    }

    if (!re.group.4) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-
Key " ", "")
        + re.group.1 + " " + re.group.2 + " " + re.group.3;
      return;
    }

    if (!re.group.5) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-
Key " ", "")
        + re.group.1 + " " + re.group.2 + " " + re.group.3 + " " + re.group.4;
      return;
    }
    set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-Ke
y " ", "")
      + re.group.1 + " " + re.group.2 + " " + re.group.3 + " " + re.group.4 + " " + re.gro
up.5;
  }
}
```

2. Call the subroutine in `vcl_fetch`:

```
sub vcl_fetch {
call construct_skey;
}
```

3. Check your success by curling an object not already in cache with the `Fastly-Debug:1` header to expose the surrogate keys. For example:

```
$ curl -svo /dev/null http://www.example.com/test/test2/file3.txt -HFastly-Debug:1
* Trying 192.0.2.0...
* Connected to www.example.com (192.0.2.0) port 80 (#0)
> Host: www.example.com
> User-Agent: curl/7.43.0
> Accept: */*
> Fastly-Debug:1
>
< HTTP/1.1 200 OK
< Server: Apache
< Content-Type: text/plain
< Surrogate-Key: /test /test/test2 /test/test2/file3.txt
< Via: 1.1 varnish
< X-Backend-IP: 203.0.113.0
< Cache-Control: max-age=31536000, stale-while-revalidate=31536000, stale-if-error=3153600
0
< Content-Length: 19
< Accept-Ranges: bytes
< Date: Fri, 29 Jan 2016 21:30:08 GMT
< Via: 1.1 varnish
< Age: 1035
< Connection: keep-alive
< Fastly-Debug-Path: (D cache-sjc3123-SJC 1454103008) (F cache-sjc3134-SJC 1454101973) (D
cache-den6026-DEN 1454101973) (F cache-den6027-DEN 1454101973)
< Fastly-Debug-TTL: (H cache-sjc3123-SJC - - 1035) (M cache-den6026-DEN - - 0)
< Fastly-Debug-Digest: b43bd38cf940e1669c2927c8662660e5170758053dda42e772ce3fc34ee57fc1
< X-Served-By: cache-den6026-DEN, cache-sjc3123-SJC
< X-Cache: MISS, HIT
< X-Cache-Hits: 0, 1
< Vary: Accept-Encoding
<
{ [19 bytes data]
* Connection #0 to host www.example.com left intact
```

In the above example, the `< Surrogate-Key: /test /test/test2 /test/test2/file3.txt` headers show the addition of the three surrogate keys.

Via the API

You can also use our key-based purging via the API to perform wildcard purging using an HTTP request:

```
POST /service/<Fastly Service ID>/purge/*/foo
Fastly-Key: FASTLY_API_TOKEN
```

This will purge any content that was associated with the `"*/foo"` surrogate key according to the setup in your header rules. Additional syntax for purging a service through the API can be found in the Purging section (/api/purge) of the API (/api/) documentation.

§ Accept-Language header VCL features (/guides/vcl/accept-language-header-vcl-features)

Fastly provides a number of extensions to VCL (/guides/vcl/guide-to-vcl#fastlys-vcl-extensions), including functions to parse and normalize the `Accept-Language` header.

Language lookup

We've implemented Lookup functionality as defined by RFC 4647, section 3.4 (http://tools.ietf.org/html/rfc4647#section-3.4).

Syntax

`accept.language_lookup(<available languages>, <default>, <priority list>)`

Argument	Explanation
`available languages`	A colon-separated list of languages to choose from. Typically the languages that the origin can provide. For example: `en:de:fr:pt:es:zh-CN`
`default`	The default language to return if none from the `priority list` match. For example: `en`
`priority list`	The `Accept-Language` header. A comma-separated list of languages, optionally accompanied by weights (q-values). For example: `pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4`

Return values

The best matching language (as per the RFC) is returned. If none are found, the default language is returned, unless a weight of zero (`q=0`) was indicated by the priority list, in which case `NULL` is returned.

Examples

```
set req.http.Normalized-Language =
  accept.language_lookup("en:de:fr:pt:es:zh-CN", "en", req.http.Accept-Language);
```

The above would result in `Normalized-Language: pt` given an `Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4` header.

`accept.language_lookup("en", "nl", "en-GB")` results in `en`, as each subtag is removed and the match retried when a tag does not match.

`accept.language_lookup("en:nl", "nl", "en-GB,nl;q=0.5")` results in `en` still, even if `nl` is a more exact match, because the q-value of nl is lower, and that has precedence. Exactness just does not come into the equation.

`accept.language_lookup("en-US:nl", "nl", "en-GB,nl;q=0.5")` results in `nl`, because subtags are not removed from the available languages, only from language tags on the priority list.

`accept.language_lookup("en-US:nl", "nl", "en-us,nl;q=0.5")` results in `en-US`, as the lookup is case insensitive.

`accept.language_lookup("en-US:nl", "nl", "en-GB,nl;q=0")` results in `NULL` (the value, not a string) since `en-GB` and `en` do not match, and `nl` (the default) is listed as unacceptable.

If `q=0` for the default language is to be ignored, the following VCL can be used:

```
set req.http.Normalized-Language =
  accept.language_lookup("en-US:nl", "nl", req.http.Accept-Language);
if (!req.http.Normalized-Language) {
  # User will get Dutch even if he doesn't want it!
  # (Because none of our languages were acceptable)
  set req.http.Normalized-Language = "nl";
}
```

Language filter (Basic)

We've implemented Basic Filtering functionality as defined by RFC 4647, section 3.3.1 (http://tools.ietf.org/html/rfc4647#section-3.3.1). The implementation is not exact when the wildcard tag (`*`) is used. If a wildcard is encountered and no matches have been found yet, the default is returned. If there are matches, those are returned and the remainder of the priority list is ignored. There is no implementation of Extended Filtering, but if you are in need you could always file a feature request with Support.

Syntax

`accept.language_filter_basic(<available languages>, <default>, <priority list>, <limit>)`

Argument	Explanation

Argument	Explanation
`available languages`	A colon-separated list of languages choose from. Typically the languages that the origin can provide. For example: `en:de:fr:pt:es:zh-CN`
`default`	The default language to return if none from the `priority list` match. For example: `en`
`priority list`	The `Accept-Language` header. A comma-separated list of languages, optionally accompanied by weights (q-values). For example: `pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4`
`limit`	The maximum amount of languages returned.

Return values

The best matching language (as per the RFC) is returned. If none are found, the default language is returned, unless a weight of zero (`q=0`) was indicated by the priority list, in which case `NULL` is returned.

Examples

```
set req.http.Filtered-Language =
  accept.language_filter_basic("en:de:fr:pt:es:zh-CN", "en", req.http.Accept-Language, 2);
```

The above would result in `Filtered-Language: pt,en` given an `Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4` header.

`accept.language_filter_basic("en", "nl", "en-GB", 2)` results in `en`, as each subtag is removed and the match retried when a tag does not match.

`accept.language_filter_basic("en:nl", "nl", "en-GB,nl;q=0.5", 2)` results in `en,nl`, even if `nl` is a more exact match, because the q-value of nl is lower, and that has precedence. Exactness just does not come into the equation.

`accept.language_filter_basic("en-US:nl", "nl", "en-GB,nl;q=0.5", 2)` results in `nl`, because subtags are not removed from the available languages during the search.

`accept.language_filter_basic("en-US:nl", "nl", "en-us,nl;q=0.5", 2)` results in `en-US,nl`, as the lookup is case insensitive.

`accept.language_filter_basic("en-US:nl", "nl", "en-GB,nl;q=0", 2)` results in `NULL` (the value, not a string) since `en-GB` and `en` do not match, and `nl` (the default) is listed as unacceptable.

If `q=0` for the default language is to be ignored, the following VCL can be used:

```
set req.http.Filtered-Language =
  accept.language_filter_basic("en-US:nl", "nl", req.http.Accept-Language, 2);
if (!req.http.Filtered-Language) {
  # User will get Dutch even if he doesn't want it!
  # (Because none of our languages were acceptable)
  set req.http.Filtered-Language = "nl";
}
```

`accept.language_filter_basic("en:nl:de:fr", "nl", "en-GB,*;q=0.5", 2)` results in `en` and `accept.language_filter_basic("en:nl:de:fr", "nl", "*", 2)` results in `nl`.

§ Authenticating before returning a request (/guides/vcl/authenticating-before-returning-a-request)

Performing authentication before returning a request is possible if your authentication is completely header-based and you do something like the following using custom VCL (/guides/vcl/uploading-custom-vcl):

```
sub vcl_recv {

  /* unset state tracking header to avoid client sending it */
  if (req.restarts == 0) {
    unset req.http.X-Authed;
  }

  if (!req.http.X-Authed) {
    /* stash the original URL and Host for later */
    set req.http.X-Orig-URL = req.url;

    /* set the URL to what the auth backend expects */
    set req.url = "/authenticate";

    /* Auth requests won't be cached, so pass */
    return(pass);
  }

  if (req.http.X-Authed == "true") {
    /* were authed, so proceed with the request */
    /* reset the URL */
    set req.url = req.http.X-Orig-URL;

  } else {
    /* the auth backend refused the request, so 403 the client */
    error 403;
  }

#FASTLY recv

  ...etc...
}

sub vcl_deliver {

  /* if we are in the auth phase */
  if (!req.http.X-Authed) {

    /* if we got a 5XX from the auth backend, we should fail open */
    if (resp.status >= 500 && resp.status < 600) {
      set req.http.X-Authed = "true";
    }

    if (resp.status == 200) {

      /* the auth backend responded with 200, allow the request and restart */
      set req.http.X-Authed = "true";
    } else if (resp.status == 401) {

      return(deliver);
```

```
    } else {

      /* the auth backend responded with non-200, deny the request and restart */
      set req.http.X-Authed = "false";
    }

    restart;
  }

#FASTLY deliver

  ...etc...
}
```

NOTE: Be sure to change `/authenticate` to whatever your authentication endpoint is.

If you feel like you can cache the authentication, then add the appropriate headers to the hash in `vcl_hash` and `return(lookup)` instead of `(pass)`.

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

§ Conditionally changing a URL (/guides/vcl/conditionally-changing-a-url)

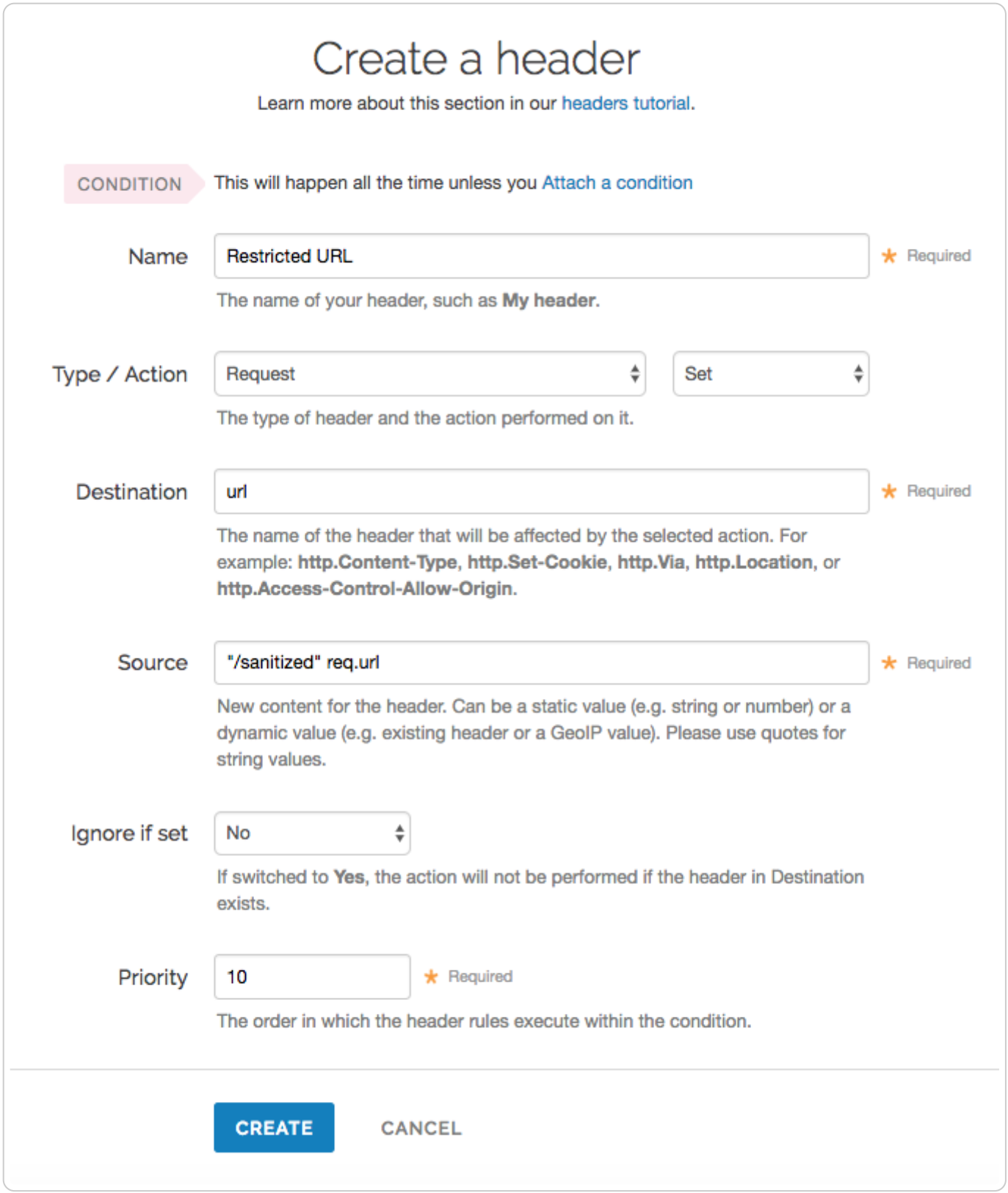
To conditionally change a URL based on the domain, include VCL that looks something like this:

```
if (req.http.host ~ "^restricted") {
  set req.url = "/sanitized" req.url;
}
```

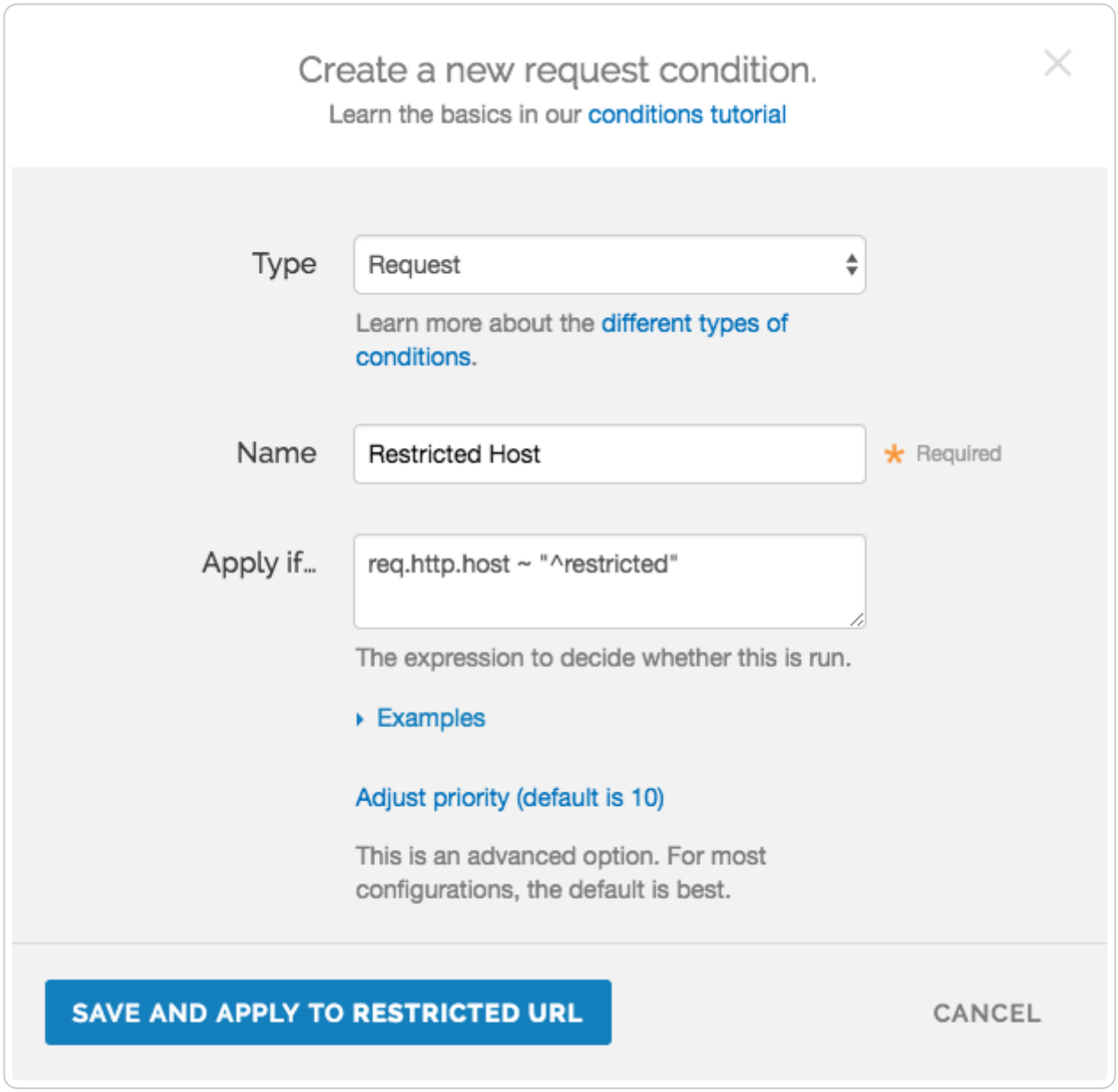
If you have shielding enabled, however, add the following code instead to avoid rewriting the URL twice:

```
if (req.http.host ~ "^restricted" && req.url !~ "^/sanitized") {
  set req.url = "/sanitized" req.url;
}
```

In Fastly's web interface, this VCL would be the equivalent of creating a new Header (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses):



and then creating a request condition that restricts connections to that host:



§ Creating location-based tagging (/guides/vcl/creating-location-based-tagging)

You can set custom HTTP headers in your varnish configuration (VCL) based on the variables we expose. Use the geolocation features we have built into Varnish to create location-based tagging. We provide a list of geographic information based on a client's IP address. For a complete list of available geolocation variables, read about which geolocation features (/guides/vcl/geolocation-related-vcl-features) are accessible via VCL.

In the example below, an HTTP header Fastly-GeoIP-CountryCode is created with the two letter country code of the client's IP address.

```
sub vcl_recv {
  ...
  set req.http.fastly-GeoIP-CountryCode = client.geo.country_code;
  ...
}
```

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

§ Cryptographic- and hashing-related VCL functions (/guides/vcl/cryptographic-and-hashing-related-vcl-functions)

Fastly provides several functions in VCL (/guides/vcl/) for cryptographic- and hashing-related purposes. It is based very heavily on Kristian Lyngstøl's digest vmod (https://github.com/varnish/libvmod-digest) for Varnish 3 (which means you can also refer to that documentation for more detail).

Functions

Function	Description
`digest.awsv4_hmac(<secret_access_key>, <yyyymmdd>, <aws_region>, <aws_service>, <string_to_sign>)`	Returns an AWSv4 message authentication code (http://docs.aws.amazon.com/AmazonS3/latest/API/v4-authenticating-requests.html#signing-request-int based on the supplied `secret_access_key` and `string_to_sign`. This function automatically prepen "AWS4" in front of the secret access key (the first function parameter) as required by the protocol. This function does not support binary data for its `secret_access_key` or `string_to_sign` parameters. See the example.
`digest.hmac_md5(<key>,<message>)`	Hash-based message authentication code (https://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using MD5. Returns a hex-encoded string prepended with 0x.

Function	Description
`digest.hmac_md5_base64(<key>, <message>)`	Hash-based message authentication code (https://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using MD5. Returns a base64-encoded (https://en.wikipedia.org/wiki/Base64) string.
`digest.hmac_sha1(<key>, <message>)`	Hash-based message authentication code (https://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using SHA1 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr Returns a hex-encoded string prepended with 0x.
`digest.hmac_sha1_base64(<key>, <message>)`	Hash-based message authentication code (https://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using SHA1 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr Returns a base64-encoded (https://en.wikipedia.org/wiki/Base64) string.
`digest.hmac_sha256(<key>, <message>))`	Hash-based message authentication code (https://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using SHA25 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr Returns a hex-encoded string prepended with 0x.
`digest.hmac_sha256_base64(<key>, <message>)`	Hash-based message authentication code (https://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using SHA25 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr Returns a base64-encoded (https://en.wikipedia.org/wiki/Base64) string.
`digest.base64(<string>)`	Base64 (https://en.wikipedia.org/wiki/Base64) encoc Returns the base64-encoded version of the input-str
`digest.base64url(<string>)`	Base64 (https://en.wikipedia.org/wiki/Base64#URL_applicati encoding. Returns the base64-encoded version of th input-string. Replaces +/ with -_ for url safety.

Function	Description
`digest.base64url_nopad(<string>)`	Base64 (https://en.wikipedia.org/wiki/Base64#URL_applicati encoding. Returns the base64-encoded version of th input-string. Replaces +/ with -_ for url safety. Has n length padding (https://en.wikipedia.org/wiki/Base64#Padding).
`digest.base64_decode(<string>)`	Decode Base64. Returns a string.
`digest.base64url_decode(<string>)`	Decode Base64 with url safe characters in. Returns string.
`digest.base64url_nopad_decode(<string>)`	Decode Base64 with url safe characters. Returns a string. Identical to base64_url_decode.
`digest.hash_sha1(<string>)`	Use the SHA1 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr hash. Returns a hex-encoded string.
`digest.hash_sha224(<string>)`	Use the SHA224 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr hash. Returns a hex-encoded string.
`digest.hash_sha256(<string>)`	Use the SHA256 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr hash. Returns a hex-encoded string.
`digest.hash_sha384(<string>)`	Use the SHA384 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr hash. Returns a hex-encoded string.
`digest.hash_sha512(<string>)`	Use the SHA512 (https://en.wikipedia.org/wiki/Secure_Hash_Algorithr hash. Returns a hex-encoded string.
`digest.hash_md5(<string>)`	Use the MD5 (https://en.wikipedia.org/wiki/MD5) has Returns a hex-encoded string.
`digest.hash_crc32(<string>)`	Use a 32 bit Cyclic Redundancy Checksum (https://en.wikipedia.org/wiki/CRC32). Returns a hex encoded string.
`digest.hash_crc32b(<string>)`	A reversed CRC32 (for compatibility with some PHP applications (http://php.net/manual/en/function.hash file.php#104836)). Returns a hex-encoded string.

Function	Description
`digest.rsa_verify(ID hash_method, STRING_LIST public_key, STRING_LIST payload, STRING_LIST digest [, ID base64_method ])`	A boolean function that returns true if the RSA diges using `public_key` of `payload` matches `digest`. The `hash_method` parameter selects the digest function t use. It can be `sha256`, `sha384`, `sha512`, or `default` (`default` is equivalent to `sha256`). The `base64_meth` parameter is optional. It can be `standard`, `url`, `url_nopad`, or `default` (`default` is equivalent to `url_nopad`).
`digest.secure_is_equal(STRING_LIST s1, STRING_LIST s2)`	A boolean function that returns true if s1 and s2 are equal. The comparison is done in constant time to defend against timing attacks.
`digest.time_hmac_md5(<base64 encoded key>, <interval>, <offset>)`	Time based One Time Password using MD5. Returns base64 encoded output.
`digest.time_hmac_sha1(<base64 encoded key>, <interval>, <offset>)`	Time based One Time Password using SHA1. Retur base64 encoded output.
`digest.time_hmac_sha256(<base64 encoded key>, <interval>, <offset>)`	Time based One Time Password using SHA256. Ret base64 encoded output.

Notes

In base64 decoding, the output theoretically could be in binary but is interpreted as a string. So if the binary output contains '\0' then it could be truncated.

The time based One-Time Password algorithm initializes the HMAC using the key and appropriate hash type. Then it hashes the message

`(<time now in seconds since UNIX epoch> / <interval>) + <offset>`

as a 64bit unsigned integer (little endian) and base64 encodes the result.

Examples

One-Time Password Validation (Token Authentication)

Use this to validate tokens with a URL format like the following:

`http://cname-to-fastly/video.mp4?6h2YUl1CB4C50SbkZ0E6U3dZGjh+84dz3+Zope2Uhik=`

Example implementations for token generation in various languages can be found in GitHub (https://github.com/fastly/token-functions).

Example VCL

```
sub vcl_recv {

  /* make sure there is a token */
  if (req.url !~ "[?&]token=([^&]+)") {
    error 403;
  }

  if (re.group.1 != digest.time_hmac_sha256("RmFzdGx5IFRva2VuIFRlc3Q=", 60, 0) &&
      re.group.1 != digest.time_hmac_sha256("RmFzdGx5IFRva2VuIFRlc3Q=", 60, -1)) {
    error 403;
  }

#FASTLY recv

  ...
}
```

Signature

```
set resp.http.x-data-sig = digest.hmac_sha256("secretkey",resp.http.x-data);
```

Base64 decoding

A snippet like this in `vcl_error` would set the response body to the value of the request header field named `x-parrot` after base64-decoding the value:

```
synthetic digest.base64_decode(req.http.x-parrot);
```

However, if the base64-decoded string contains a NUL byte (0x00), then that byte and any bytes following it will not be included in the response. Keep that in mind if you intend to send a synthetic response that contains binary data. There is currently no way to send a synthetic response containing a NUL byte.

AWSv4 message authentication

Use the following format:

```
STRING awsv4_hmac(STRING key, STRING date_stamp, STRING region, STRING service, STRING string)
```

Example VCL

```
set resp.http.sig = digest.awsv4_hmac(
    "wJalrXUtnFEMI/K7MDENG+bPxRfiCYEXAMPLEKEY",
    "20120215",
    "us-east-1",
    "iam",
    "hello");
```

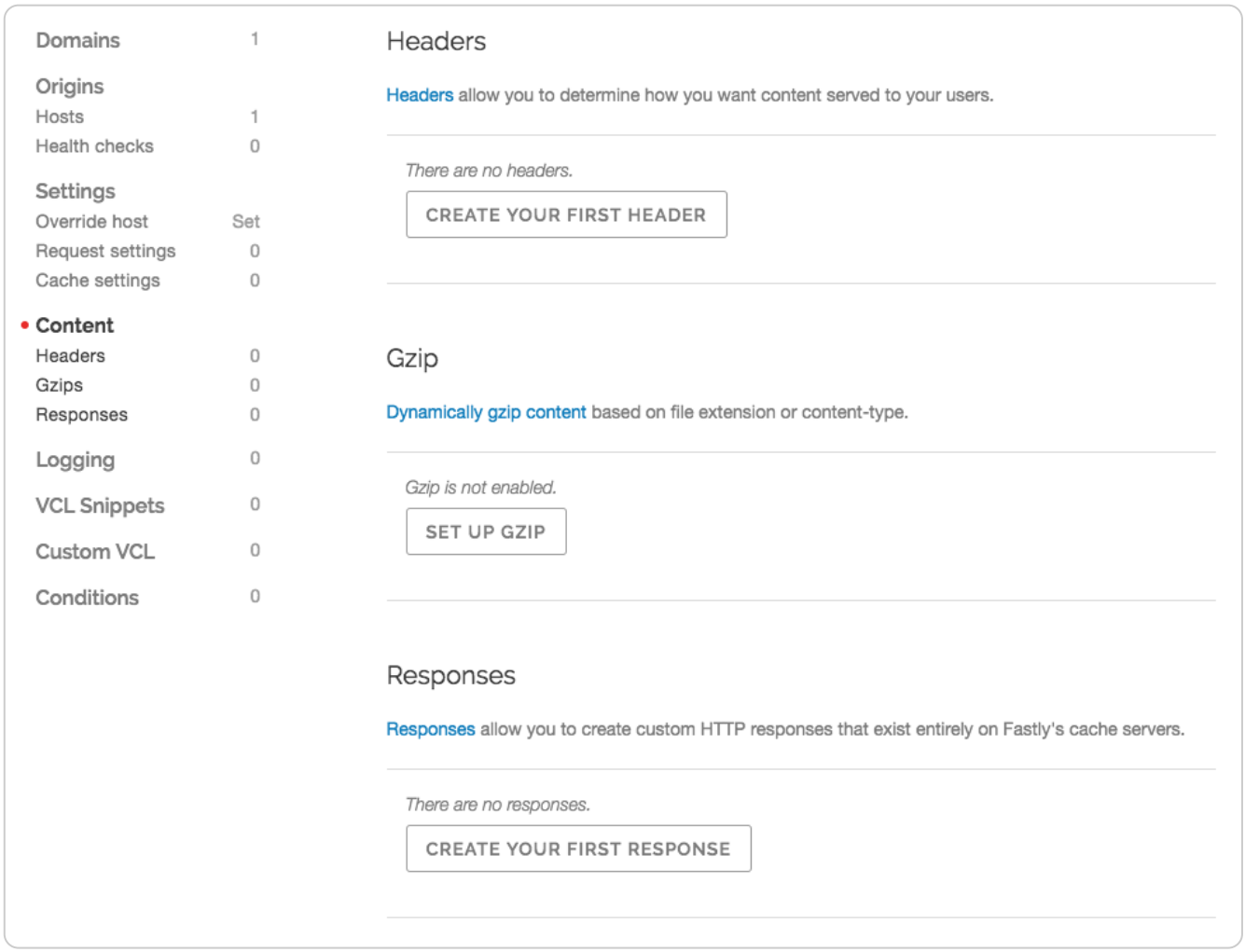
§ Custom responses that don't hit origin servers (/guides/vcl/custom-responses-that-dont-hit-origin-servers)

Fastly can send custom responses for certain requests that you don't want to hit your origin servers. For example, if you wanted to restrict caching to a URL subtree that contains images and scripts, you could configure Fastly to return an `HTTP 404 Not Found` response to requests for anything other than `/Content/*` or `/Scripts/*`. To illustrate how to implement this example, we'll show you how to create a response and corresponding request condition.

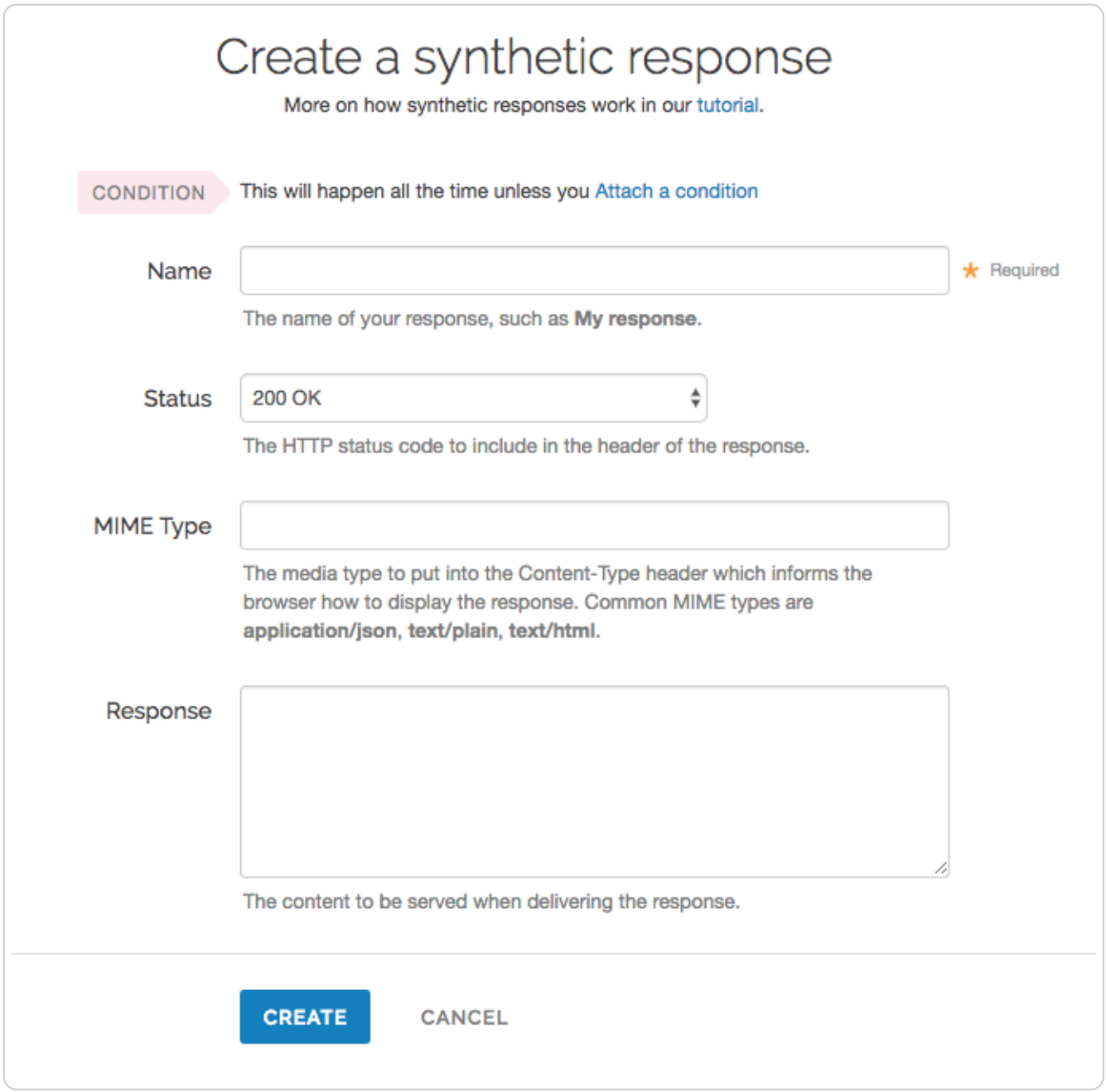
Creating the response

Follow these instructions to create a new response for your service:

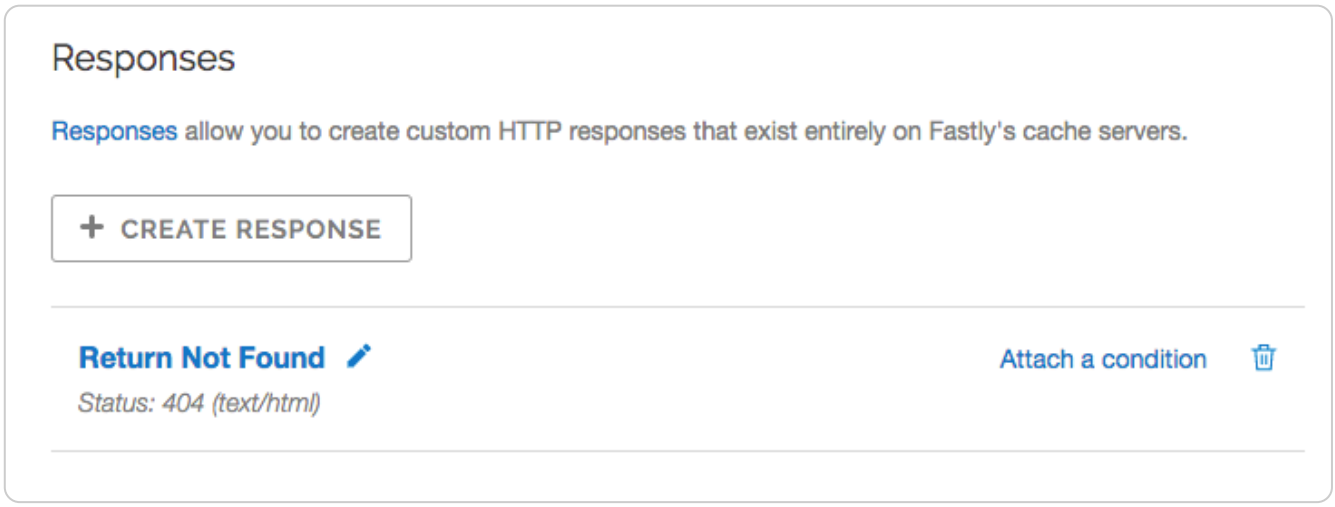
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create response** button. The Create a synthetic response page appears.



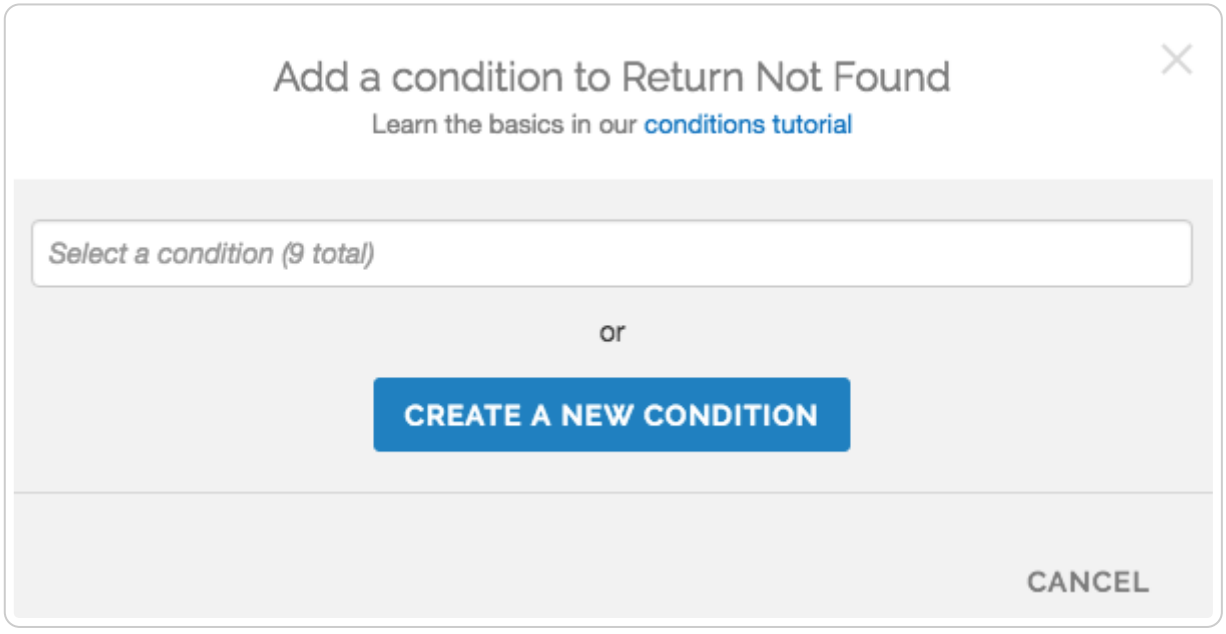
6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type a human-readable name for the response. For example `Return Not Found`.
   - From the **Status** menu, select an HTTP code to return to the client. For example, `404 Not Found`.
   - In the **MIME Type** field, type the MIME type of the response. For example, `text/html`.
   - In the **Response** field, type the plaintext or HTML content to return to the client. For example `Page not found`.
7. Click the **Create** button to create the response. The new response appears in the Responses area of the Content page.



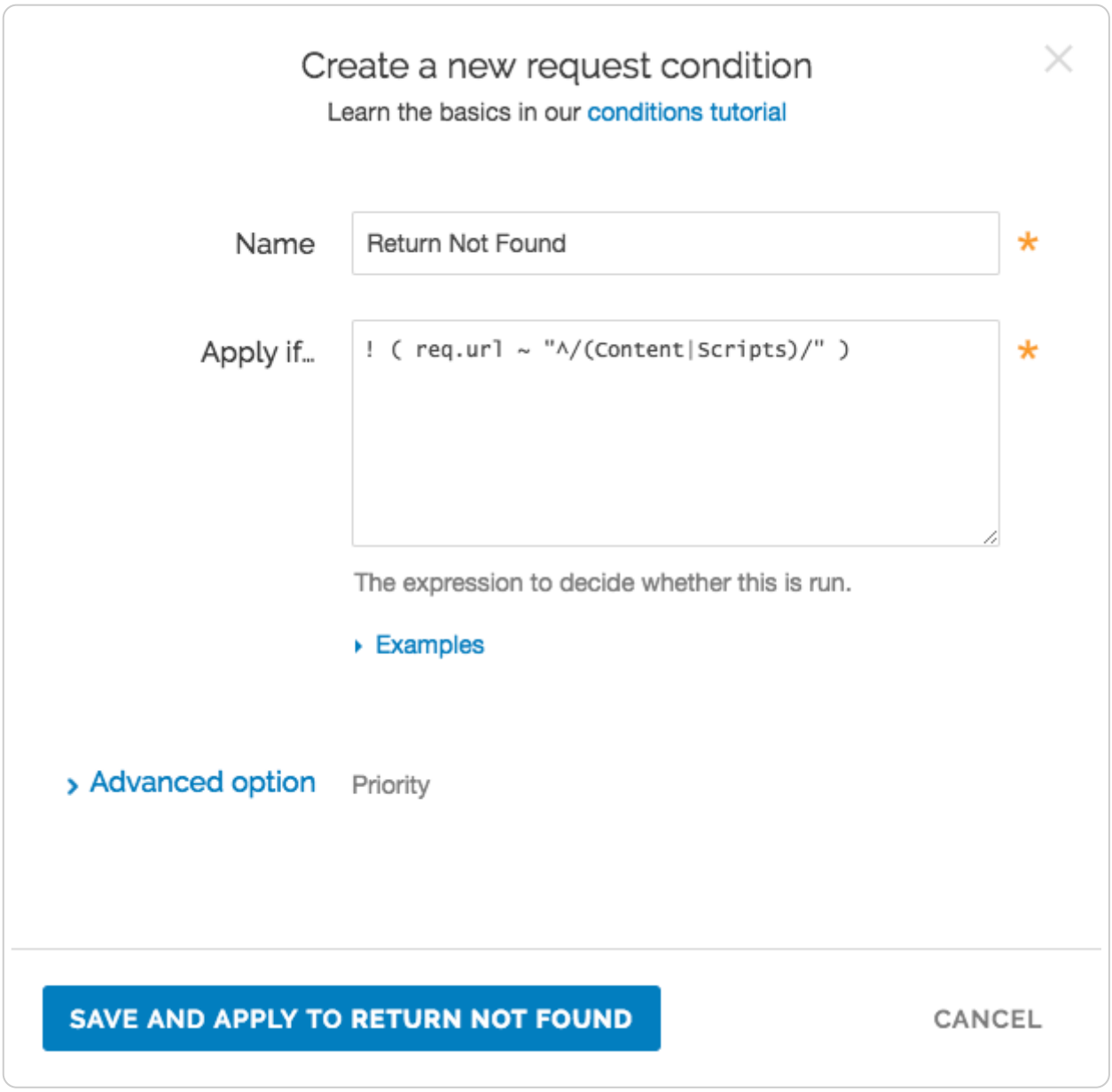
Creating the request condition

Follow these instructions to attach a request condition to the response you just created:

1. Click the **Attach a condition** link next to the response that you just created. The Add a condition window appears.



2. Click the **Create a new condition** button. The Create a new condition window appears.



3. Fill out the **Create a new condition** fields as follows:
   - In the **Name** field, type a human-readable name for the condition. For example, `Return Not Found`.
   - In the **Apply if** field, type the request condition you want inserted into a VCL if statement. For example, `! ( req.url ~ "^/(Content|Scripts)/" )`. See below for more examples of request conditions.
4. Click the **Save and apply to** button. The Responses area now displays the condition that must be met in order for your response to begin being used.



5. Click the **Activate** button to deploy your configuration changes.

Example request conditions

Respond only if URLs don't match a certain mask, in this case `/Content/*` or `/Scripts/*`:

```
! (req.url ~ "^/(Content|Scripts)/")
```

Respond only if URLs match `/secret/*` or are Microsoft Word or Excel documents (`*.doc` and `*.xls` file extensions):

```
! (req.url ~ "^/secret/" || req.url ~ "\.(xls|doc)$")
```

Ignore POST and PUT HTTP requests:

```
req.request == "POST" || req.request == "PUT"
```

Deny a spider or crawler using `user-agent` `"annoying_robot"`:

```
req.http.user-agent ~ "annoying_robot"
```

Prevent a specific IP from connecting, in this case the IP `225.0.0.1`:

```
client.ip == "225.0.0.1"
```

Use geographic variables (/guides/vcl/geolocation-related-vcl-features) to block traffic from a specific location (e.g., China):

```
client.geo.country_code == "CN"
```

Match the `client.ip` against a CIDR range, such as `240.24.0.0/16` (this requires first creating an ACL object in VCL (/guides/vcl/using-access-control-lists)):

```
client.ip ~ ipRangeObject
```

§ Date- and time-related VCL features (/guides/vcl/date-and-time-related-vcl-features)

By default VCL includes the `now` variable, which provides the current time (for example, `Wed, 17 Sep 2025 23:19:06 GMT`). Fastly adds several new Varnish variables and functions (/guides/vcl/) that allow more flexibility when dealing with dates and times.

Variables

The following variables have been added:

Name	Location	Description
`now`	all	The current time in RFC 1123 format (https://www.ietf.org/rfc/rfc1123.txt) format.
`now.sec`	all	Like the `now` variable, but in seconds since the UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time).
`time.start`	all	The time the request started, after TLS termination, using RFC 1123 format (https://www.ietf.org/rfc/rfc1123.txt).
`time.start.sec`	all	The time the request started in seconds since the UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time), after TLS termination.
`time.start.msec`	all	The time the request started in milliseconds since the UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time), after TLS termination.
`time.start.msec_frac`	all	The time the request started in milliseconds since the last whole second, after TLS termination.
`time.start.usec`	all	The time the request started in microseconds since the UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time), after TLS termination.

Name	Location	Description
`time.start.usec_frac`	all	The time the request started in microseconds since the last whole second, after TLS termination.
`time.end`	deliver, log	The time the request ended, using RFC 1123 format (https://www.ietf.org/rfc/rfc1123.txt). Also useful for strftime.
`time.end.sec`	deliver, log	The time the request ended in seconds since the UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time).
`time.end.msec`	deliver, log	The time the request ended in milliseconds since the UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time).
`time.end.msec_frac`	deliver, log	The time the request started in milliseconds since the last whole second.
`time.end.usec`	deliver, log	The time the request ended in microseconds since the UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time).
`time.end.usec_frac`	deliver, log	The time the request started in microseconds since the last whole second.
`time.elapsed`	deliver, log	The time since the request start, using RFC 1123 format (https://www.ietf.org/rfc/rfc1123.txt). Also useful for strftime.
`time.elapsed.sec`	deliver, log	The time since the request start in seconds.
`time.elapsed.msec`	deliver, log	The time since the request start in milliseconds.
`time.elapsed.msec_frac`	deliver, log	The time the request started in milliseconds since the last whole second.
`time.elapsed.usec`	deliver, log	The time since the request start in microseconds.
`time.elapsed.usec_frac`	deliver, log	The time the request started in microseconds since the last whole second

Name	Location	Description
`time.to_first_byte`	deliver, log	The time interval since the request started up to the point before the `vcl_deliver` function ran. When used in a string context, an RTIME variable like this one will be formatted as a number in seconds with 3 decimal digits of precision. In `vcl_deliver` this interval will be very close to `time.elapsed`. In `vcl_log`, the difference between `time.elapsed` and `time.to_first_byte` will be the time that it took to send the response body.

Functions

The following functions have been added:

Name	Description
`time.hex_to_time(<divider>, <hextime>)`	Takes a hexadecimal string value, divides by `divider` and interprets the result as seconds since UNIX Epoch (https://en.wikipedia.org/wiki/Unix_time).
`time.add(<time>, <offset>)`	Adds `offset` to `time`.
`time.sub(<time>, <offset>)`	Subtracts `offset` from `time`.
`time.is_after(<time1>, <time2>)`	Returns `TRUE` if `time1` is after `time2`. (Normal timeflow and causality required.)
`strftime(<format>, <time>)`	Formats a time to a string. This uses standard POSIX strftime formats (http://www.unix.com/man-page/FreeBSD/3/strftime/). See the examples section for information about handling conflicting encoding of `%` characters.
`std.time(<string_to_parse>, <fallback_value>)`	Converts a string to a time variable. The following string formats are supported: `Sun, 06 Nov 1994 08:49:37 GMT`, `Sunday, 06-Nov-94 08:49:37 GMT`, `Sun Nov  6 08:49:37 1994`, `784111777.00`, `784111777`. Useful because time variables are needed as arguments for functions like `time.add` and `strftime`.

Name	Description
`std.integer2time(<seconds_since_epoch>)`	Converts an integer to a time variable. To use a string, use `std.atoi`.

Examples

The following examples illustrate how to use the variables and functions.

★ **TIP:** Regular strings ("short strings") in VCL use `%xx` escapes (percent encoding) for special characters, which would conflict with the `%` used in the strftime format. For the strftime examples, we use VCL "long strings" `{"..."}`, which do not use the `%xx` escapes. Alternatively, you could use `%25` for each `%`.

time.hex_to_time, time.add, and time.is_after

```
if (time.is_after(time.add(now, 10m), time.hex_to_time(1, "d0542d8"))) {
  ...
}
```

strftime

```
set resp.http.Now = strftime({"%Y-%m-%d %H:%M"}, now)

set resp.http.Start = strftime({"%a, %d %b %Y %T %z"}, time.start)
```

std.time and std.integer2time

```
set resp.http.X-Seconds-Since-Modified = strftime({"%s"}, time.sub(now, std.time(resp.http.Last
-Modified, now)));

std.integer2time(std.atoi("1445445162"));
```

Comparison operators like `> < >= <= == !=` do not work with `std.time` or `std.integer2time`. Instead, you can compare two times using something similar to this:

```
if (time.is_after(now, std.integer2time(std.atoi("1445445162")))) {
  # do something
}
```

§ Delivering different content to different devices (/guides/vcl/delivering-different-content-to-different-devices)

The easiest way to deliver different content based on the device being used is to rewrite the URL of the request based on what the user agent is. We've written an article that describes how to change the URL based on conditions (/guides/vcl/conditionally-changing-a-url) using our user interface but in pure VCL it would look something like this:

```
sub vcl_recv {
  if (req.http.User-Agent ~ "(?i)ip(hone|od)") {
    set req.url = "/mobile" req.url;
  } elsif (req.http.User-Agent ~ "(?i)ipad") {
    set req.url = "/tablet" req.url;
  }
#FASTLY recv
}
```

Obviously the code fragment above doesn't contain a comprehensive list of mobile and tablet devices. Google has an official blog post (https://webmasters.googleblog.com/2011/03/mo-better-to-also-detect-mobile-user.html) on detecting Android mobile versus tablet and this VCL fragment (https://github.com/varnishcache/varnish-devicedetect/blob/master/devicedetect.vcl) from Varnish Software can detect several different types of devices quite reliably, although it doesn't include Windows mobile and tablet, Blackberry Playbook, and the Kindle user agents.

The most comprehensive device detection routine we've seen so far is this one:

```
# based on https://github.com/varnish/varnish-devicedetect/blob/master/devicedetect.vcl
sub detect_device {
  unset req.http.X-UA-Device;
  unset req.http.X-UA-Vendor;

  set req.http.X-UA-Device = "desktop";
  set req.http.X-UA-Vendor = "generic";

  # Handle that a cookie or url param may override the detection altogether
  if (req.url ~ "[&|?]device_force=([^&\s]+)") {
    set req.http.X-UA-Device = regsub(req.url, ".*[&|?]device_force=([^&\s]+).*", "\1");
  } elsif (req.http.Cookie ~ "(?i)X-UA-Device-force") {
    # ;?? means zero or one ;, non-greedy to match the first
    set req.http.X-UA-Device = regsub(req.http.Cookie, "(?i).*X-UA-Device-force=([^;]+);??.*",
"\1");
    # Clean up our mess in the cookie header
    set req.http.Cookie = regsuball(req.http.Cookie, "(^|; ) *X-UA-Device-force=[^;]+;? *",
"\1");
    # If the cookie header is now empty, or just whitespace, unset it
    if (req.http.Cookie ~ "^ *$") { unset req.http.Cookie; } # "$ # stupid syntax highlighter
  } else {
    if (req.http.User-Agent ~ "(?i)(ads|google|bing|msn|yandex|baidu|ro|career|)bot" ||
      req.http.User-Agent ~ "(?i)(baidu|jike|symantec)spider" ||
      req.http.User-Agent ~ "(?i)scanner" ||
      req.http.User-Agent ~ "(?i)(web)crawler") {
      set req.http.X-UA-Device = "bot";
    } elsif (req.http.User-Agent ~ "(?i)ipad") {
      set req.http.X-UA-Device = "tablet";
      set req.http.X-UA-Vendor = "apple";
    } elsif (req.http.User-Agent ~ "(?i)ip(hone|od)") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "apple";
    # how do we differ between an android phone and an android tablet?
    #   http://stackoverflow.com/questions/5341637/how-do-detect-android-tablets-in-general-use
ragent
    #   http://googlewebmastercentral.blogspot.com/2011/03/mo-better-to-also-detect-mobile-use
r.html
    } elsif (req.http.User-Agent ~ "(?i)android.*(mobile|mini)") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "android";
    # android 3/honeycomb was just about tablet-only, and any phones will probably handle a big
ger page layout
    } elsif (req.http.User-Agent ~ "(?i)android") {
      set req.http.X-UA-Device = "tablet";
      set req.http.X-UA-Vendor = "android";
    # see http://my.opera.com/community/openweb/idopera/
    } elsif (req.http.User-Agent ~ "Opera Mobi") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "android";
    } elsif (req.http.User-Agent ~ "PlayBook; U; RIM Tablet") {
      set req.http.X-UA-Device = "tablet";
```

```
  set req.http.X-UA-Vendor = "blackberry";
} elsif (req.http.User-Agent ~ "hp-tablet.*TouchPad") {
  set req.http.X-UA-Device = "tablet";
  set req.http.X-UA-Vendor = "hp";
} elsif (req.http.User-Agent ~ "Kindle/3") {
  set req.http.X-UA-Device = "tablet";
  set req.http.X-UA-Vendor = "kindle";
} elsif (req.http.User-Agent ~ "Mobile.+Firefox") {
  set req.http.X-UA-Device = "mobile";
  set req.http.X-UA-Vendor = "firefoxos";
} elsif (req.http.User-Agent ~ "^HTC") {
  set req.http.X-UA-Device = "smartphone";
  set req.http.X-UA-Vendor = "htc";
} elsif (req.http.User-Agent ~ "Fennec") {
  set req.http.X-UA-Device = "smartphone";
  set req.http.X-UA-Vendor = "fennec";
} elsif (req.http.User-Agent ~ "IEMobile") {
  set req.http.X-UA-Device = "smartphone";
  set req.http.X-UA-Vendor = "microsoft";
} elsif (req.http.User-Agent ~ "BlackBerry" || req.http.User-Agent ~ "BB10.*Mobile") {
  set req.http.X-UA-Device = "smartphone";
  set req.http.X-UA-Vendor = "blackberry";
} elsif (req.http.User-Agent ~ "GT-.*Build/GINGERBREAD") {
  set req.http.X-UA-Device = "smartphone";
  set req.http.X-UA-Vendor = "android";
} elsif (req.http.User-Agent ~ "SymbianOS.*AppleWebKit") {
  set req.http.X-UA-Device = "smartphone";
  set req.http.X-UA-Vendor = "symbian";
} elsif (req.http.User-Agent ~ "(?i)symbian" ||
  req.http.User-Agent ~ "(?i)^sonyericsson" ||
  req.http.User-Agent ~ "(?i)^nokia" ||
  req.http.User-Agent ~ "(?i)^samsung" ||
  req.http.User-Agent ~ "(?i)^lg" ||
  req.http.User-Agent ~ "(?i)bada" ||
  req.http.User-Agent ~ "(?i)blazer" ||
  req.http.User-Agent ~ "(?i)cellphone" ||
  req.http.User-Agent ~ "(?i)iemobile" ||
  req.http.User-Agent ~ "(?i)midp-2.0" ||
  req.http.User-Agent ~ "(?i)u990" ||
  req.http.User-Agent ~ "(?i)netfront" ||
  req.http.User-Agent ~ "(?i)opera mini" ||
  req.http.User-Agent ~ "(?i)palm" ||
  req.http.User-Agent ~ "(?i)nintendo wii" ||
  req.http.User-Agent ~ "(?i)playstation portable" ||
  req.http.User-Agent ~ "(?i)portalmmm" ||
  req.http.User-Agent ~ "(?i)proxinet" ||
  req.http.User-Agent ~ "(?i)sonyericsson" ||
  req.http.User-Agent ~ "(?i)symbian" ||
  req.http.User-Agent ~ "(?i)windows\ ?ce" ||
  req.http.User-Agent ~ "(?i)winwap" ||
  req.http.User-Agent ~ "(?i)eudoraweb" ||
```

```
      req.http.User-Agent ~ "(?i)htc" ||
      req.http.User-Agent ~ "(?i)240x320" ||
      req.http.User-Agent ~ "(?i)avantgo") {
      set req.http.X-UA-Device = "mobile";
    }
  }
}
```

§ Geolocation-related VCL features (/guides/vcl/geolocation-related-vcl-features)

Fastly exposes a number of geographic variables for you to take advantage of inside VCL (/guides/vcl/guide-to-vcl) for both IPv4 and IPv6 client IPs. These variables appear as follows:

Variable	Description
`client.geo.latitude`	The latitude associated with the IP address.
`client.geo.longitude`	The longitude associated with the IP address.
`client.geo.city`	The city or town name associated with the IP address, encodec the ASCII character encoding (a lowercase ASCII approximatior original string with diacritics removed).
`client.geo.city.ascii`	An alias of `client.geo.city`.
`client.geo.city.utf8`	The city or town name associated with the IP address, encodec the UTF-8 character encoding.
`client.geo.continent_code`	A two-character code representing the continent associated wit IP address. Possible codes are: **AF** - Africa, **AS** - Asia, **EU** - Eur **NA** - North America, **OC** - Oceania, **SA** - South America, **AN** - Antarctica.
`client.geo.country_code`	A two-character ISO 3166-1 (https://en.wikipedia.org/wiki/ISO_ 1) country code for the country associated with the IP address. **US** country code is returned for IP addresses associated with overseas United States military bases.

Variable	Description
`client.geo.country_code3`	A three-character ISO 3166-1 alpha-3 (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3) country cod the country associated with the IP address. The **USA** country c returned for IP addresses associated with overseas United Stat military bases.
`client.geo.country_name`	The country name associated with the IP address, encoded usi ASCII character encoding (a lowercase ASCII approximation of original string with diacritics removed).
`client.geo.country_name.ascii`	An alias of `client.geo.country_name`.
`client.geo.country_name.utf8`	The country name associated with the IP address, encoded usi UTF-8 character encoding.
`client.geo.postal_code`	The postal code associated with the IP address. These are avai for some IP addresses in Australia, Canada, France, Germany, I Spain, Switzerland, the United Kingdom, and the United States. return the first 3 characters for Canadian postal codes. We retu first 2-4 characters (outward code) for postal codes in the Unite Kingdom.
`client.geo.region`	The ISO 3166-2 (https://en.wikipedia.org/wiki/ISO_3166-2) regi code associated with the IP address.
`client.geo.area_code`	The telephone area code associated with the IP address. These only available for IP addresses in the United States.
`client.geo.metro_code`	The metro code associated with the IP address. Metro codes represent designated market areas (DMAs) in the United States Germany, Independent Television Service (ITV) regions in the UK department codes in France, South Korean administrative divisi (si, gun, gu or cities, counties, and districts), Chinese administra regions (diji shi, or "region-level" cities), Russian federal districts Norwegian municipalities, urban areas in New Zealand, and the Greater Capital City Statistical Area (GCCSA) and Significant Ur Areas (SUAs) in Australia.
`client.geo.gmt_offset`	The time zone offset from coordinated universal time (UTC) for t `client.geo.city` associated with the IP address.

Variable	Description
`client.geo.conn_speed`	The type of connection speed (http://www.webopedia.com/quick_ref/internet_connection_typ associated with the IP address. Possible values are: **broadbanc cable**, **dialup**, **mobile**, **oc12**, **oc3**, **t1**, **t3**, **satellite**, **wireless**, **xd**
`client.as.number`	The autonomous system (AS) (https://en.wikipedia.org/wiki/Autonomous_system_(Internet)) n associated with this IP address.
`client.as.name`	The name of the organization associated with `client.as.numbe`

NOTE: Geolocation information, including data streamed by our log streaming service (/guides/streaming-logs/), is intended to be used only in connection with your use of Fastly services. Use of geolocation data for other purposes may require the permission of a IP geolocation dataset vendor, such as Digital Element (http://www.digitalelement.com/end-user-license-agreement-eula/).

Fastly also exposes codes that describe the location of the datacenter the request came through as follows:

Variable	Description
`server.region`	A code representing the general region of the world in which the POP location resides. One of: APAC, Asia, EU-Central, EU-East, EU-West, Middle-East, North-America, SA-East, South-Africa, South-America, US-Central, US-East, US-West.
`server.datacenter`	A code representing one of Fastly's POP locations (/guides/basic-concepts/fastly-pop-locations).

TIP: If you're updating your configurations from older version of the geolocation variables, be sure to read our migration guide (/guides/migrations/migrating-geolocation-variables-to-the-new-dataset).

Using geographic variables with shielding

If you have shielding (/guides/performance-tuning/shielding) enabled, you should set the following variable before using geographic variables:

```
set client.geo.ip_override = req.http.Fastly-Client-IP;
```

§ Guide to VCL (/guides/vcl/guide-to-vcl)

About Varnish and why Fastly uses it

Varnish is the open source software (https://www.fastly.com/open-source) Fastly commercialized with performance and capacity (among other) enhancements. Fastly's Varnish is based on Varnish 2.1 and our Varnish syntax is specifically compatible with Varnish 2.1.5. The principal configuration mechanism of Varnish software is the Varnish Configuration Language (VCL), the scripting language used to configure and add logic to Varnish caches. Varnish allows Fastly to apply changes to the cache software as it is executing. Specifically, VCL is generated, compiled, transmitted to all Fastly caches, loaded into the operating software, and activated immediately, with no waiting for maintenance windows and no server downtime.

Fastly generates VCL automatically per your specifications via the web interface (/guides/basic-concepts/about-the-web-interface-controls). We allow you to create your own VCL files with specialized configurations. Your custom VCL files can be uploaded (/guides/vcl/uploading-custom-vcl) into Fastly caches and activated. You can also mix and match (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl) custom VCL and Fastly VCL, using them together at the same time. You will never lose the options on the Fastly user interface when you use custom VCL, but keep in mind that custom VCL always takes precedence over any VCL generated by the user interface. Be mindful of where your custom VCL sits in the default VCL.

❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Fastly's VCL Extensions

In addition, Fastly has included a number of extensions to VCL that won't be covered by any other documentation. Specifically:

Extension	**Description**
accept-language features (/guides/vcl/accept-language-header-vcl-features)	Provides functions to parse and normalize the `Accept-Language` header.

Extension	Description
cryptographic and hashing functions (/guides/vcl/cryptographic-and-hashing-related-vcl-functions)	Supports Hash-based Message Authentication Code (HMAC), a message authentication code that uses a cryptographic key in conjunction with a hash function.
date- and time-related features (/guides/vcl/date-and-time-related-vcl-features)	Supports the default VCL "now" variable that provides the current time as an RFC 850 formatted date (e.g., Tuesday, 29-Apr-14 08:41:55), as well as several new functions that allow you to have more flexibility when dealing with dates and times.
Geolocation features (/guides/vcl/geolocation-related-vcl-features)	Provides the ability to search a geolocation database for a given host or IP address, and return information about the country, city or Internet Service Provider (ISP) for that IP address.
local variables in VCL (/guides/vcl/local-variables-in-vcl)	Supports variables for storing temporary values during request processing
randomness features (/guides/vcl/randomness-related-vcl-features)	Supports the insertion of random strings, content cookies, and decisions into requests.
size-related variables (/guides/vcl/size-related-vcl-variables)	Supports reporting variables that offer insight into what happened in a request.
TLS and HTTP/2 variables (/guides/vcl/tls-and-http2-vcl-variables-and-functions)	Supports the use of variables and functions related to TLS and HTTP/2.
miscellaneous features and variables (/guides/vcl/miscellaneous-VCL-extensions)	Provides miscellaneous VCL extensions not easily grouped into other categories.

Embedding inline C code in VCL

Currently, we don't provide embedded C access to our users. Fastly is a shared infrastructure. By allowing the use of inline C code, we could potentially give a single user the power to read, write to, or write from everything. As a result, our varnish process (i.e., files on disk, memory of the varnish user's processes) would become unprotected because inline C code opens the potential for users to do things like crash servers, steal data, or run a botnet.

We appreciate feedback from our customers. If you are interested in a feature that requires C code, contact support@fastly.com (mailto:support@fastly.com). Our engineering team looks forward to these kinds of challenges.

Where to learn more about Varnish and VCL

The official Varnish documentation (https://www.varnish-cache.org/docs/2.1/tutorial/vcl.html) is a good place to start when looking for online information. In addition, Varnish Software, who provides commercial support for Varnish, has written a free online book (http://info.varnish-software.com/the-varnish-book).

Roberto Moutinho's book *Instant Varnish Cache (http://www.amazon.com/Instant-Varnish-Cache-Roberto-Moutinho/dp/178216040X)* also provides information.

---

§ Isolating header values without regular expressions (/guides/vcl/isolating-header-values-without-regular-expressions)

Fastly supports the ability to extract header subfield values without regular expressions in a human-readable way.

"Headers subfields" refer to headers with a body syntax style similar to `value1=123value123; testValue=asdf_true; loggedInTest=true;` or `max-age=0, surrogate-control=3600` These headers include Cookie, Set-Cookie, Cache-Control, or a custom header. Fastly allows you to isolate these key values with the following syntax:

```
req.http.Header-Name:key-name
```

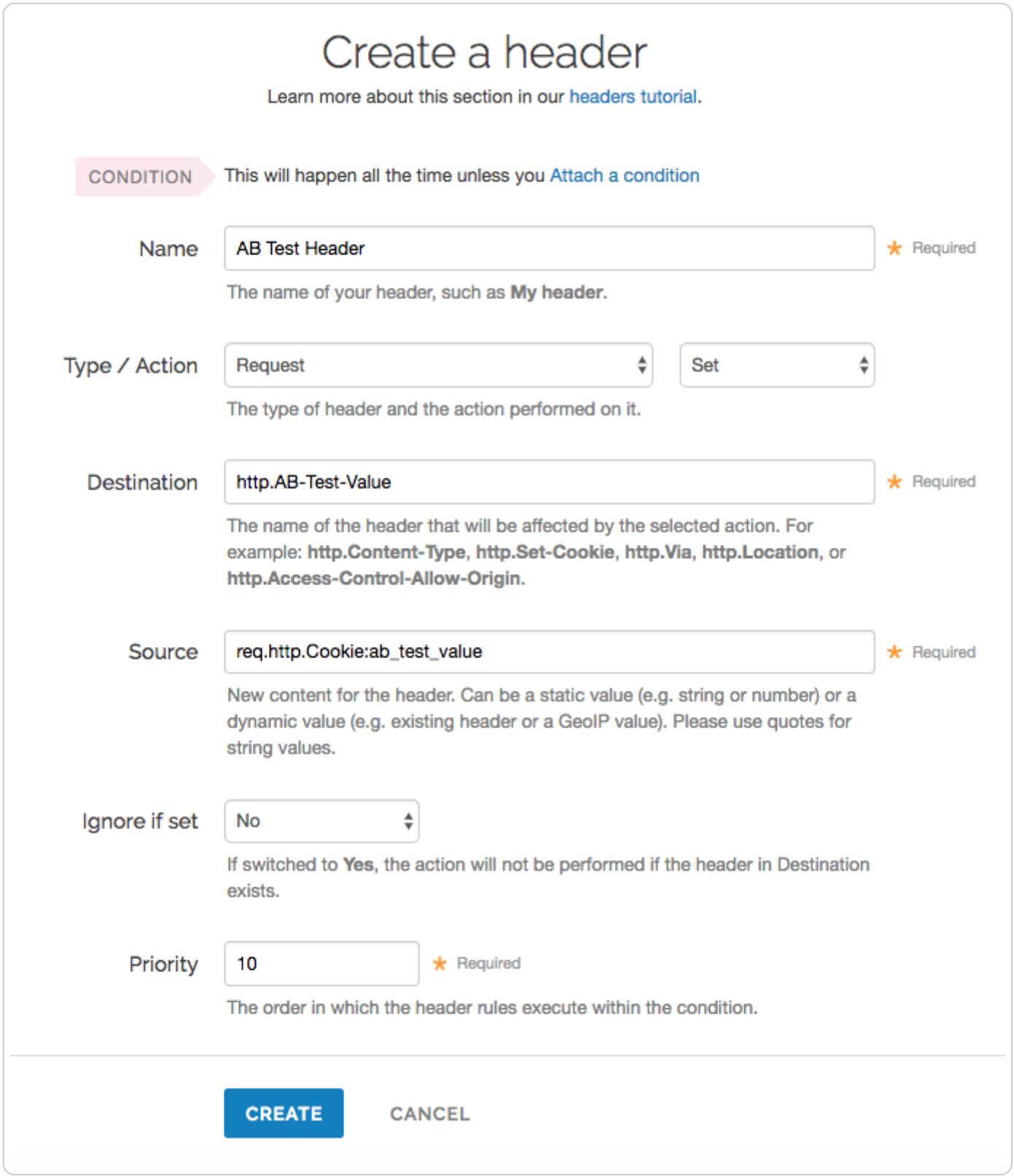
In cases where a `Set-Cookie` response from origin is `value1=123value123; testValue=asdf_true; loggedInTest=true;`, the code for isolating the `loggedInTest` value would be:

```
beresp.http.Set-Cookie:loggedInTest
```

This logic can be used in uploaded custom VCL, as well as throughout the web interface. For example, using VCL this logic would execute based on the value of staff_user within req.http.Cookie.

```
# in vcl_recv
if (req.http.Cookie:staff_user ~ "true") {
  # some logic goes here
  return(pass);
}
```

For example, to isolate the value of `ab_test_value` from `Cookie` to the header `req.http.AB-Test-Value`, create a custom header (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) with the following settings:



Fill out the **Create a header** fields as follows:

- In the **Name** field, type `AB Test Header`.
- From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
- In the **Destination** field, type `http.AB-Test-Value`.
- In the **Source** field, type `req.http.Cookie:ab_test_value`.

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

This will send the `AB-Test-Value` header in every inbound request.

---

§ Local variables in VCL (/guides/vcl/local-variables-in-vcl)

Fastly VCL (/guides/vcl/guide-to-vcl) supports variables for storing temporary values during request processing.

Declaring a variable

Variables must be declared before they are used, usually at the beginning of the function before any statements. They can only be used in the same function where they are declared. Variables with the same name in different function blocks are unrelated variables.

Variables start with `var.` and their names consist of characters in the set `[A-Za-z0-9._-]`. (`:` is explicitly disallowed.) The declaration syntax is:

```
declare local var.<name> <type>;
```

Variable types

Variables can be of the following types:

- `BOOL`
- `INTEGER`
- `FLOAT`
- `TIME` (absolute time)
- `RTIME` (relative time)
- `STRING`

Declared variables are initialized to the zero value of the type:

- `0` for numeric types
- `false` for `BOOL`
- `NULL` for `STRING`

Use

Boolean variables

Boolean assignments support boolean variables on the right-hand side as well as `BOOL`-returning functions, conditional expressions, and the `true` and `false` constants. For example:

```
declare local var.boolean BOOL;

# BOOL assignment with RHS variable
set var.boolean = true;
set req.esi = var.boolean;
set resp.http.Bool = if(req.esi, "y", "n");

# BOOL assignment with RHS function
set var.boolean = http_status_matches(resp.status, "200,304");

# BOOL assigment with RHS conditional
set var.boolean = (req.url == "/");

# non-NULL-ness check, like 'if (req.http.Foo) { ... }'
set var.boolean = (req.http.Foo);
```

Numeric variables

Numeric assignment and comparison support numeric variables (anything except `STRING` or `BOOL`) on the right-hand side, including conversion in both directions between `FLOAT` and `INTEGER` types, rounding to the nearest integer in the `FLOAT` to `INTEGER` case. For example:

```
declare local var.integer INTEGER;
declare local var.float FLOAT;

# Numeric assignment with RHS variable and
# implicit string conversion for header
set var.integer = req.bytes_read;
set var.integer -= req.body_bytes_read;
set resp.http.VarInteger = var.integer;

# Numeric comparison with RHS variable
set resp.http.VarIntegerOK = if(req.header_bytes_read == var.integer, "y", "n");
```

Invalid conditions or domain errors like division by 0 will set `fastly.error`.

String variables

String assignments support string concatenation on the right-hand side. For example:

```
declare local var.restarted STRING;

# String concatenation on RHS
set var.restarted = "Request " if(req.restarts > 0, "has", "has not") " restarted.";
```

Time variables

Time variables support both relative and absolute times. For example:

```
declare local var.time TIME;
declare local var.rtime RTIME;

set req.grace = 72s;
set var.rtime = req.grace;
set resp.http.VarRTime = var.rtime;

set var.time = std.time("Fri, 10 Jun 2016 00:02:12 GMT", now);
set var.time -= var.rtime;
# implicit string conversion for header
set resp.http.VarTime = var.time;
```

§ Manipulating the cache key (/guides/vcl/manipulating-the-cache-key)

Redefining the cache key

❗ **WARNING:** By default, Fastly uses the URL and the host of a request (plus a special, internal Fastly variable for purging (/guides/purging/single-purges) purposes) to create unique HTTP objects. Although Fastly allows you to explicitly set the cache key to define this more precisely, changing the default behavior risks the following:

1. If you add too much information to the cache key, you can significantly reduce your hit ratio.
2. If you make a mistake when explicitly setting the cache key, you can cause all requests to get the same object.
3. If you add anything to the hash, you will need to send a purge for each combination of the URL and value you add in order to purge that specific information from the cache.

To avoid these dangers, consider using the Vary header (https://www.fastly.com/blog/best-practices-for-using-the-vary-header) instead of following the instructions below.

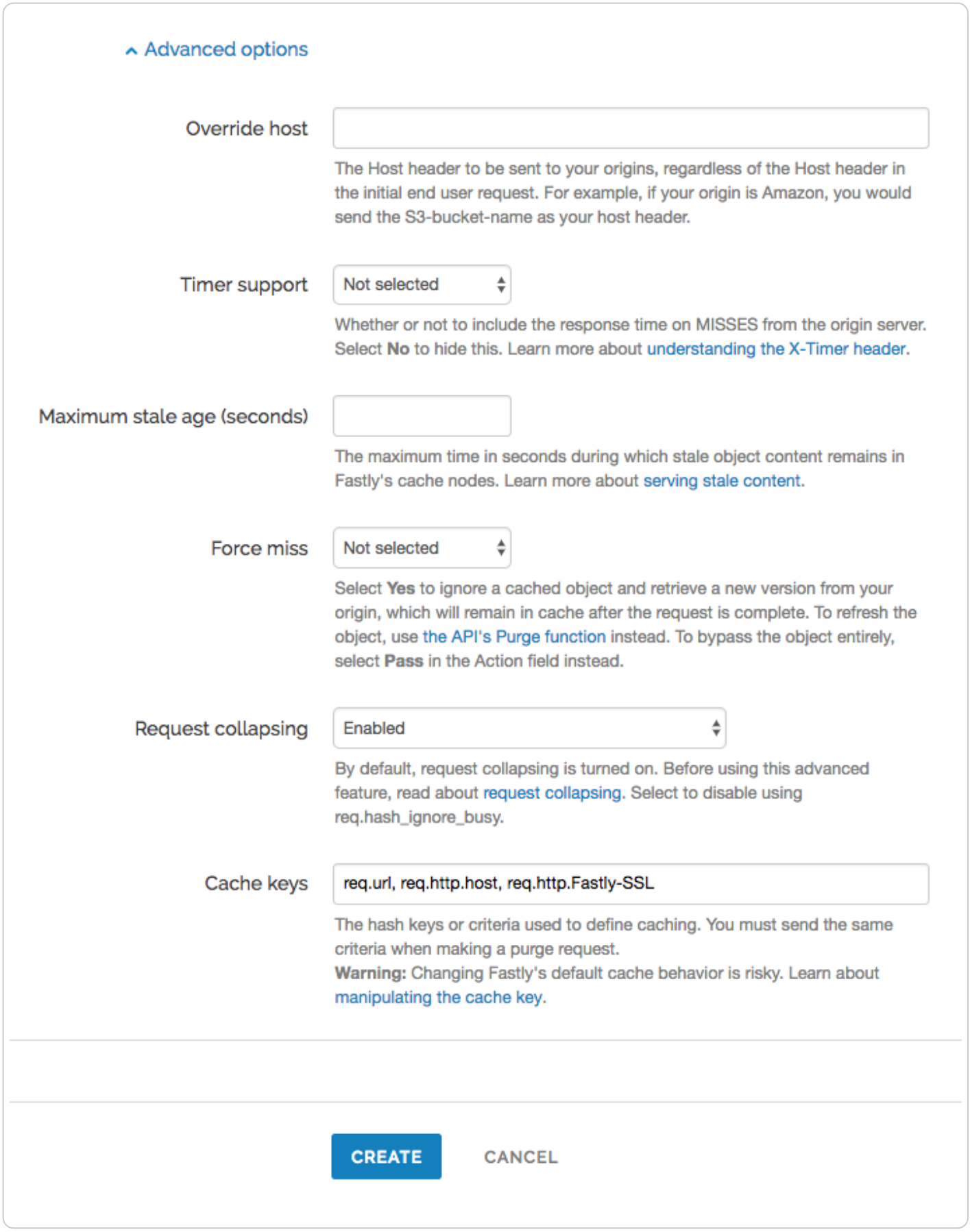
Explicitly setting the cache key

You can set the cache key explicitly (including attaching conditions (/guides/conditions/using-conditions)) by adding a request setting via the Settings page in the configuration controls (/guides/basic-concepts/about-the-web-interface-controls#about-the-configure-page) and including a comma-separated list of cache keys. The values of the cache keys listed are combined to make a single hash, and each unique hash is considered a unique object.

For example, if you don't want the query string to be part of the cache key, but you don't want to change `req.url` so that the query string still ends up in your logs, you could use the following text for the hash keys:

```
req.url.path, req.http.host
```

In the user interface, the text would appear in the Cache keys field:



As a general rule, you should always have `req.url` as one of your cache keys or as part of one.

Purging adjustments when making additions to cache keys

Because purging works on individual hashes, additions to cache keys can complicate purging URLs. However, it can also be simplified.

For example, if you were to change your cache key to just `req.url` and not the default `req.url, req.http.host`, then purging `http://foo.example.com/file.html` would also purge `http://bar.example.com/file.html`. Keep in mind this is because they're actually the same object in the cache!

On the other hand, if you were to change your cache key `req.url, req.http.host, req.http.Fastly-SSL`, you would have to purge `http://example.com/` and `https://example.com/` individually.

In the latter case, if you were to use the Vary header instead of changing the cache key, you could still have different content on the two URLs, yet purge them with a single purge. In this case you would add a new Cache Header (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses), use `http.Vary` as the Destination, and use the following as the Source:

```
if(beresp.http.Vary, beresp.http.Vary ",", "") "Fastly-SSL"
```

Using a POST request body as a cache key

As long as the body of a POST request is less than 2K in size and the content type is `application/x-www-form-urlencoded`, then we allow you to use it as part of the cache key. Your VCL (/guides/vcl/) should look something like:

```
sub vcl_hash {
  set req.hash += req.url;
  set req.hash += req.http.host;
  if (req.request == "POST" && req.postbody) {
    set req.hash += req.postbody;
  }
#FASTLY hash
  return(hash);
}
```

You'll also need to force a cache lookup, but only for requests that can be cached, by doing something like this in **vcl_recv**:

```
  if (req.request == "POST" && req.postbody ~ "(^|&)action=list(&|$)") {
    return(lookup);
  }
```

★ **TIP:** To refine this, you could add only the important parts of `req.postbody` to the hash using `regsub()`.

Using a cookie as a cache key

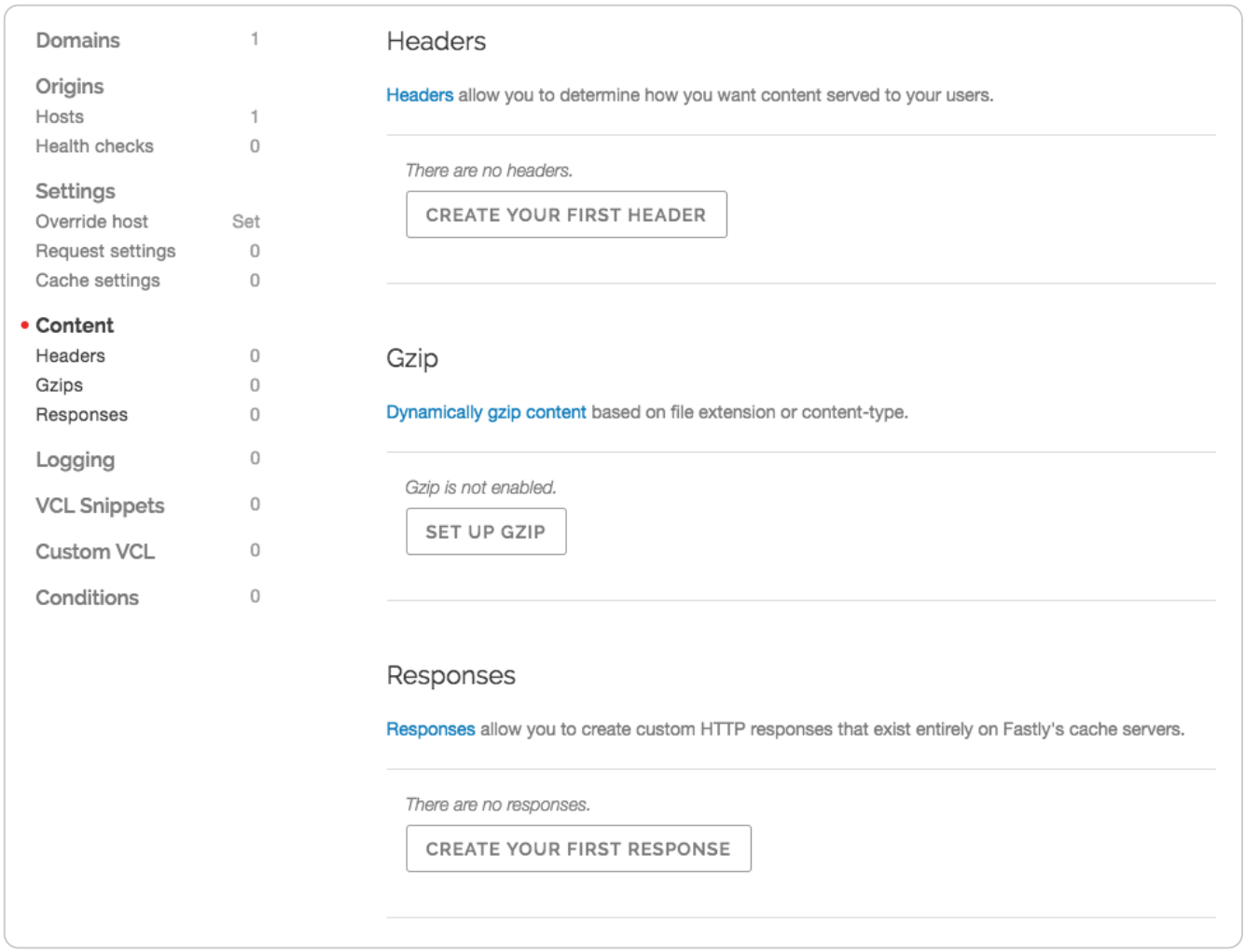
You can use a cookie as a cache key or just check for the presence of a cookie set to a specific value by controlling its request conditions (/guides/conditions/). Both methods are simple and shown in the steps below.

To use a cookie as a cache key

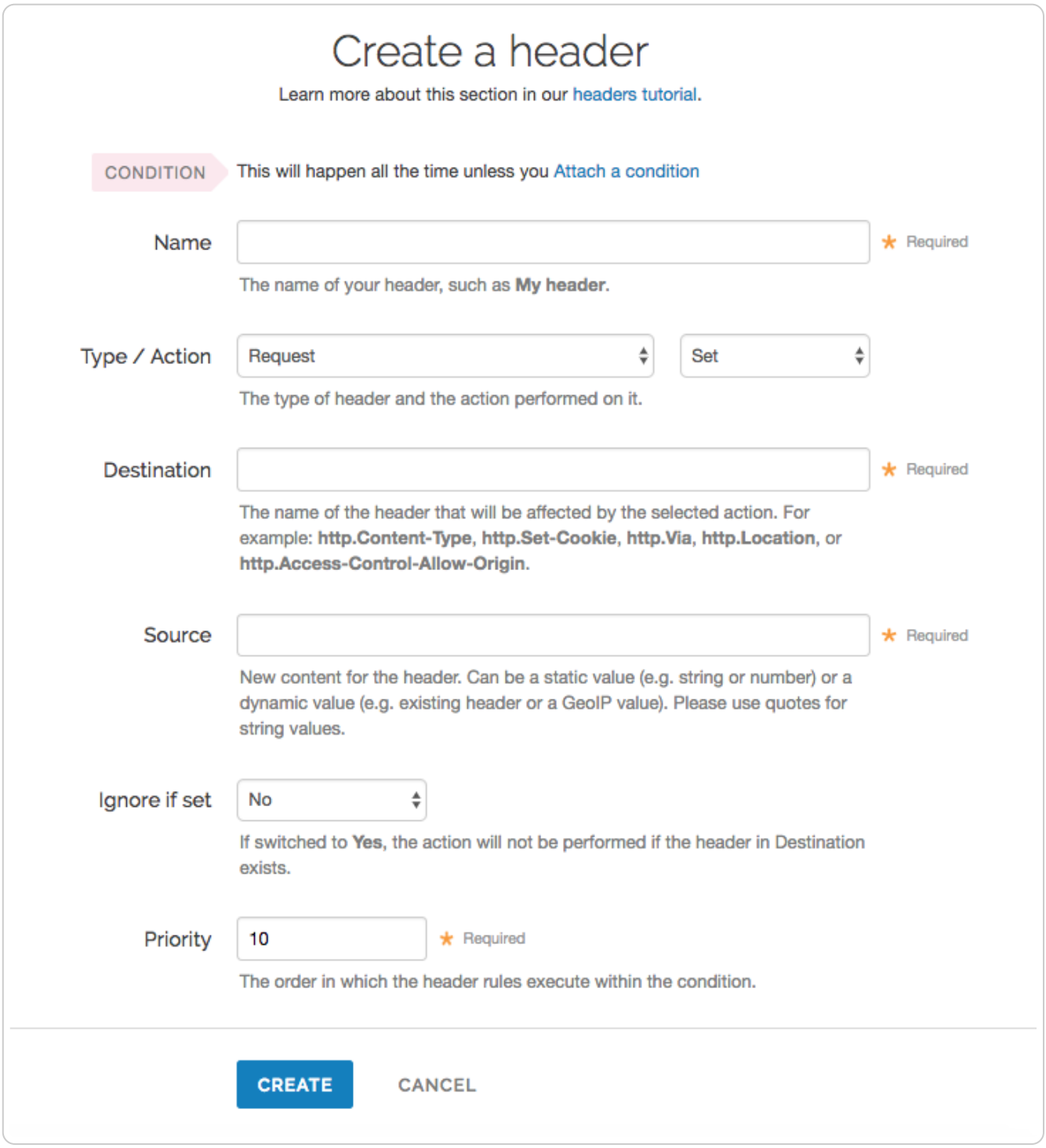
Using a cookie as a cache key looks complicated but it's actually quite simple. Let's say your cookie is called "MyCookie" and it looks like `mycookie=`

Create new headers

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



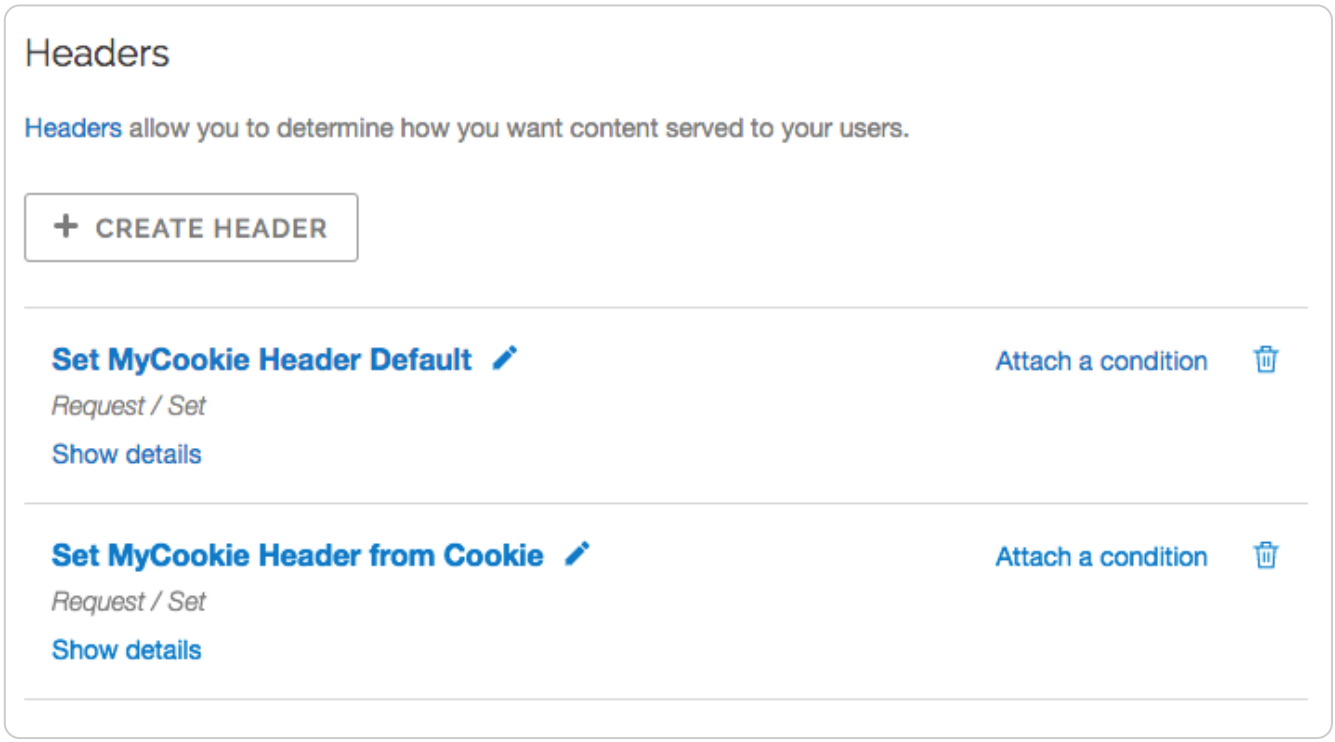
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Set MyCookie Header Default`.
   - From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
   - In the **Destination** field, type `http.X-MyCookie`.
   - In the **Source** field, type `"0"` (with quotes).
   - Leave the **Ignore if set** menu set to the default, **No**.

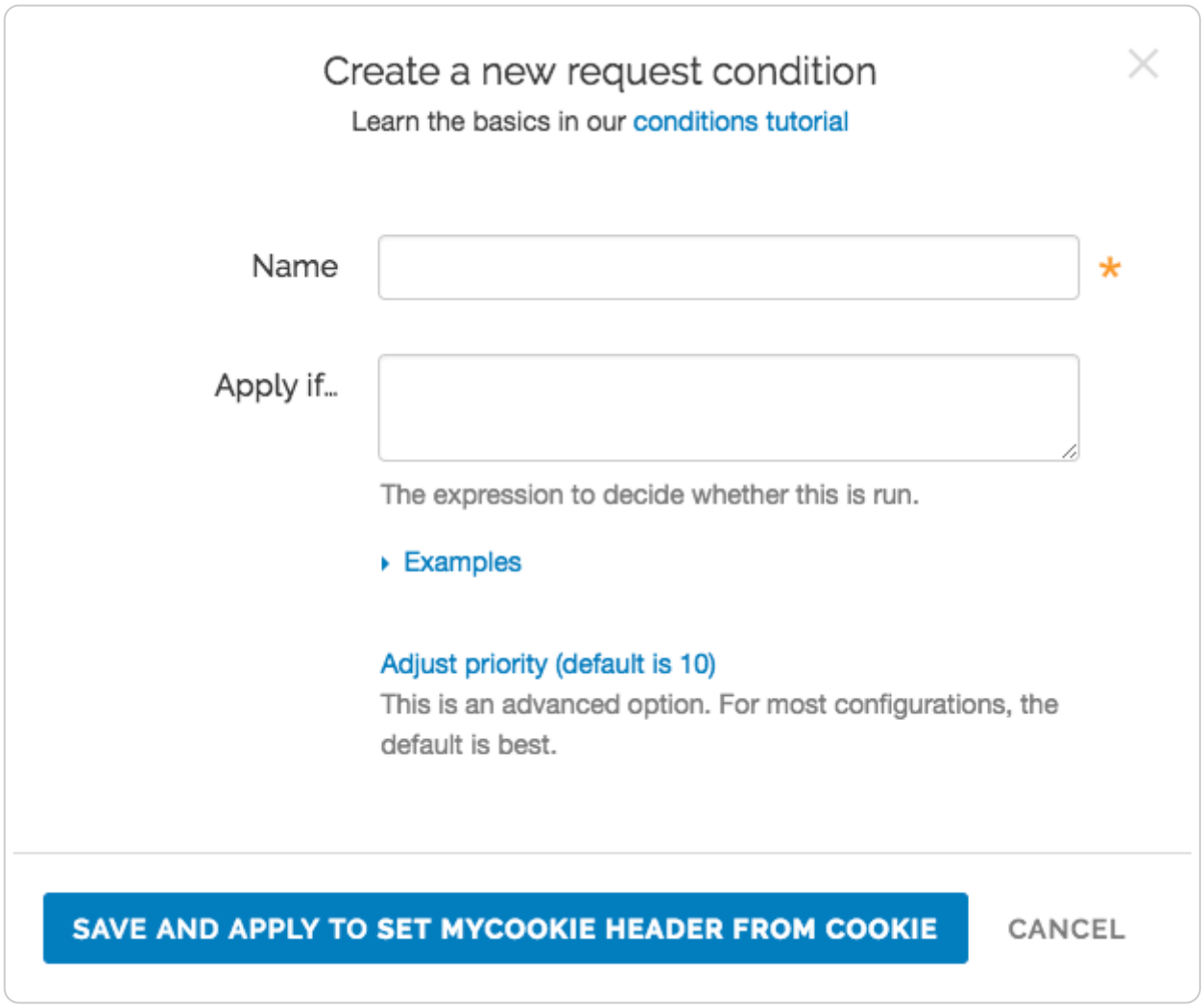
- In the **Priority** field, type a number representing the order in which the header rule should execute. The default is set to `10` for new headers.

7. Click the **Create** button. The new header appears in the Headers area of the Content page.
8. Click the **Create header** button again and create a second new header by filling out the fields as follows:
   - In the **Name** field, type `Set MyCookie Header from Cookie`.
   - From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
   - In the **Destination** field, type `http.X-MyCookie`.
   - In the **Source** field, type `req.http.cookie:mycookie`.
   - Leave the **Ignore if set** menu set to the default, **No**.
   - In the **Priority** field, type a larger number than the priority of previous header you just created. For example, if you left the default priority set to `10`, type `20`.
9. Click the **Create** button. The second header appears in the Headers area of the Content page.

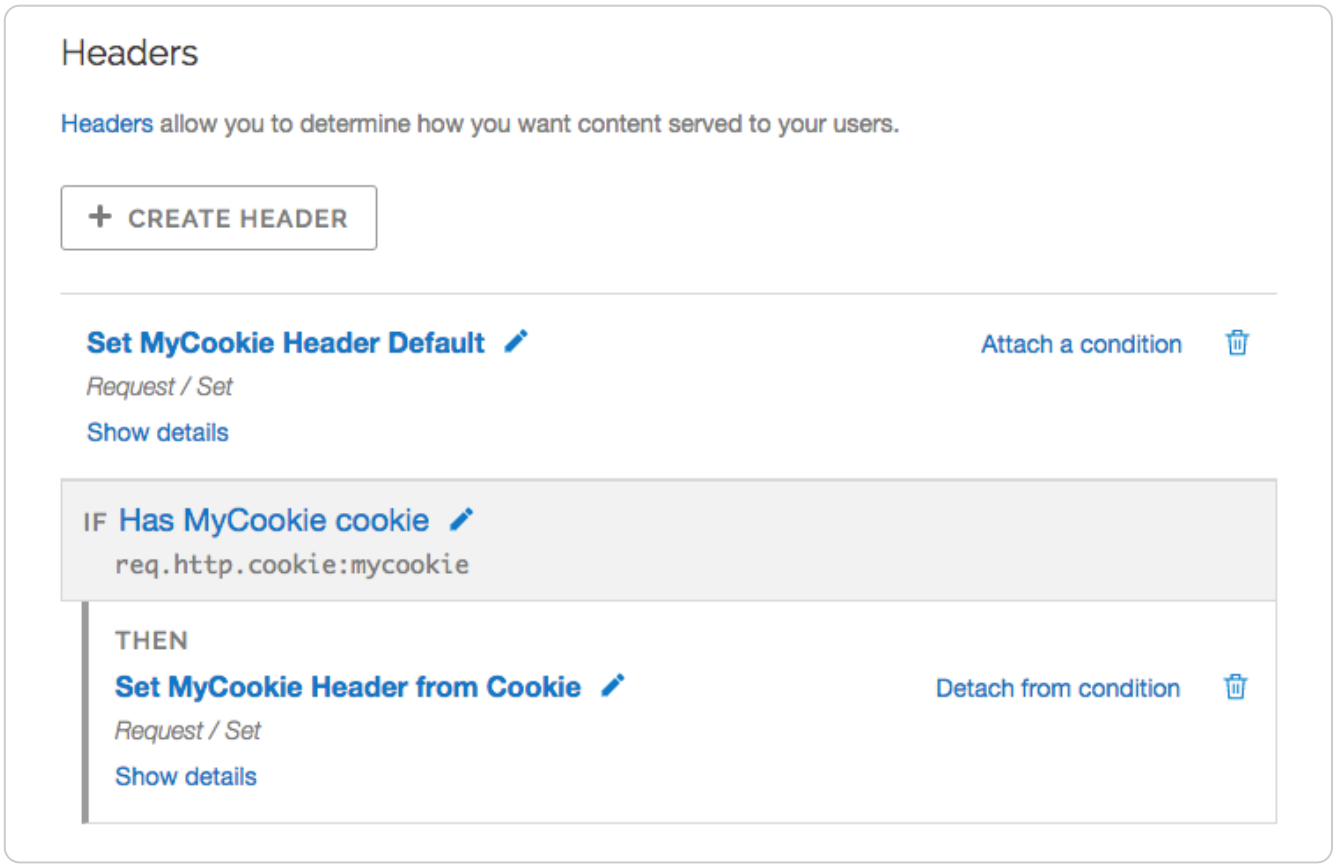


Attach conditions to the new headers

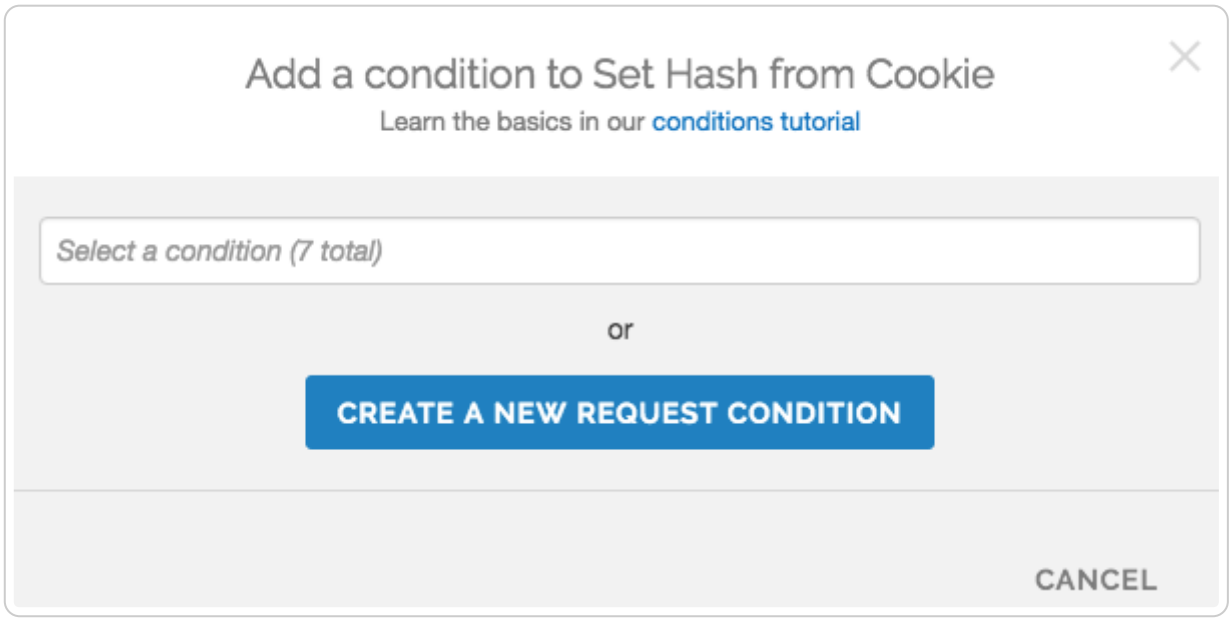
1. Click the **Attach a condition** link next to the `Set MyCookie Header from Cookie` header. The add a condition window appears.
2. Click the **Create a new request condition** button. The Create a new request condition window appears.



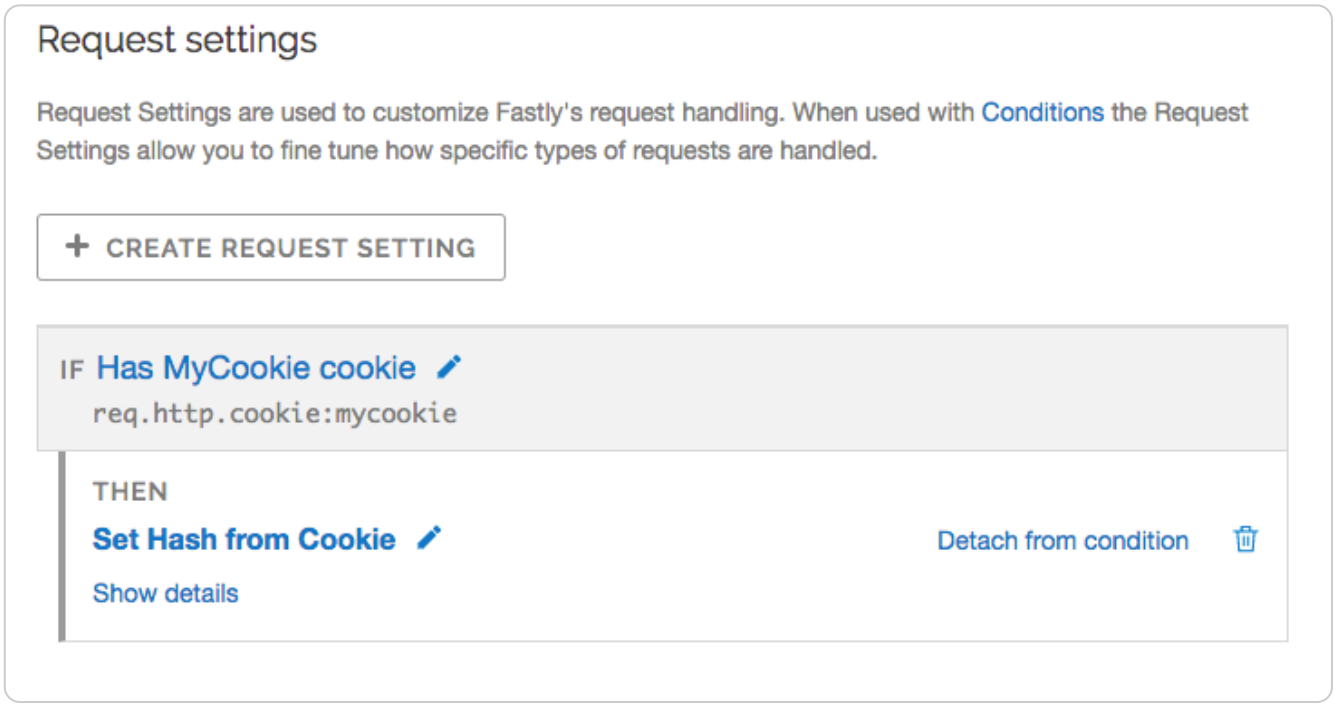
3. Fill out the fields of the **Create a new request condition** page as follows:
   - In the **Name** field, type `Has MyCookie cookie`.
   - In the **Apply if** field, type `req.http.cookie:mycookie`.
4. Click the **Save and apply to** button. The Headers area now displays the condition that must be met in order for your header to begin being used.



5. Click the **Settings** link. The Settings page appears.
6. Click the **Create request setting** button. The Create a request setting page appears.
7. In the **Name** field, type `Set Hash from Cookie`.
8. Click the **Advanced options** link. The Advanced options appear.
9. In the **Cache keys** field, type `req.url, req.http.host, req.http.X-MyCookie`
10. Click the **Create** button. The new request appears in the Request settings area.
11. Click the **Attach a condition** link next to the new request. The Add a condition window appears.



12. From the **Select a condition** menu, select `Has MyCookie cookie`. The Request settings area now displays the condition that must be met in order for your request to begin being used.



13. Click the **Activate** button to deploy your configuration changes.

To check for the presence of a cookie set to a specific value

An alternative way if you're just checking for the presence of the cookie set to some specific value (e.g., 1):

1. Add a new Request setting where the Cache key field is set to `req.url, req.http.host, "Has mycookie"`.

2. Add a condition to that Request setting where the Apply if field contains `req.http.cookie:mycookie`.

§ Miscellaneous VCL extensions (/guides/vcl/miscellaneous-VCL-extensions)

Fastly has added several miscellaneous features (/guides/vcl/guide-to-vcl) to Varnish that don't easily fit into specific categories (/guides/vcl/).

Variables

Variable	Description
`bereq.url`	The URL sent to the backend. Does not include the host and scheme, meaning in `www.example.com/index.html`, `bereq.url` would contain `/index.html`.
`bereq.url.qs`	The query string portion of `bereq.url`. This will be from immediately after the `?` to the end of the URL.
`bereq.url.basename`	Same as `req.url.basename`, except for use between Fastly and your origin servers.
`bereq.url.dirname`	Same as `req.url.dirname`, except for use between Fastly and your origin servers.
`beresp.backend.ip`	The IP of the backend this response was fetched from (backported from Varnish 3).
`beresp.backend.name`	The name of the backend this response was fetched from (backported from Varnish 3).
`beresp.backend.port`	The port of the backend this response was fetched from (backported from Varnish 3).
`beresp.grace`	Defines how long an object can remain overdue and still have Varnish consider it for grace mode. Fastly has implemented `stale-if-error` (/guides/performance-tuning/serving-stale-content#usage) as a parallel implementation of `beresp.grace`.

Variable	Description
`beresp.pci`	Specifies that content be cached in a manner that satisfies PCI DSS requirements. See our PCI compliance description (/guides/detailed-product-descriptions/pci-compliant-caching) for instructions on enabling this feature for your account.
`beresp.hipaa`	Specifies that content not be cached in non-volatile memory to help customers meet HIPAA security requirements. See our guide on HIPAA and caching PHI (/guides/compliance/hipaa-and-caching-phi) for instructions on enabling this feature for your account. An alias of `beresp.pci`.
`client.port`	Returns the remote client port. This could be useful as a seed that returns the same value both in an ESI and a top level request. For example, you could hash `client.ip` and `client.port` to get a value used both in ESI and the top level request.
`req.grace`	Defines how long an object can remain overdue and still have Varnish consider it for grace mode.
`req.http.host`	The full host name, without the path or query parameters. For example, in the request `www.example.com/index.html?a=1&b=2`, `req.http.host` will return `www.example.com`.
`req.is_ipv6`	Indicates whether the request was made using IPv6 or not. This is a boolean, read-only variable available in `vcl_recv`, `vcl_hash`, `vcl_deliver` and `vcl_log`.
`req.restarts`	Counts the number of times the VCL has been restarted.
`req.topurl`	In an ESI subrequest, returns the URL of the top-level request.
`req.url.basename`	The file name specified in a URL. For example, in the request `www.example.com/1/hello.gif?foo=bar`, `req.url.basename` will return `hello.gif`.
`req.url.dirname`	The directories specified in a URL. For example, in the request `www.example.com/1/hello.gif?foo=bar`, `req.url.dirname` will return `/1`. In the request `www.example.com/5/inner/hello.gif?foo=bar`, `req.url.dirname` will return `/5/inner`.
`req.url.ext`	The file extension specified in a URL. For example, in the request `www.example.com/1/hello.gif?foo=bar`, `req.url.ext` will return `gif`.

Variable	Description
`req.url.path`	The full path, without any query parameters. For example, in the request `www.example.com/index.html?a=1&b=2`, `req.url.path` will return `/index.html`.
`req.url`	The full path, including query parameters. For example, in the request `www.example.com/index.html?a=1&b=2`, `req.url` will return `/index.html?a=1&b=2`.
`req.url.qs`	The query string portion of `req.url`. This will be from immediately after the `?` to the end of the URL.
`stale.exists`	Specifies if a given object has stale content (/guides/performance-tuning/serving-stale-content) in cache. Returns `TRUE` or `FALSE`.

Functions

Function	Format	Description
`boltsort.sort`	`STRING boltsort.sort(STRING url)`	Sorts URL parameters. For example, `boltsort.sort("/foo?b=1&a=2&c=3");` returns `"/foo?a=2&b=1&c=3"`. Example: `set req.url = boltsort.sort(req.url);`
`cstr_escape`	`STRING cstr_escape(STRING cstr)`	Escapes characters unsafe for printing from a string. For example, `"` becomes `\"`. Example: `log "escaped=" cstr_escape("%22" req.protocol "%22")`

Function	Format	Description
`http_status_matches`	`BOOL http_status_matches(INT status, STRING fmt)`	Determines whether or not an HTTP status code matches a pattern. The arguments are an integer (usually `beresp.status` or `resp.status`) and a comma-separated list of status codes, optionally prefixed by a `!` to negate the match. It returns `TRUE` or `FALSE`. For example, `if (http_status_matches(beresp.status, "!200,404")) {`. Example: `if (http_status_matches(beresp.status, "!200,301,302")) { set obj.cacheable = 0; }`
`if()`	`STRING if(BOOL expression, STRING valueiftrue, STRING valueiffalse)`	Implements a ternary operator for strings; if the expression is true, it returns `TRUE`; if the expression is false, it returns `FALSE`. For example, you have an `if(x, true-expression, false-expression)`; if this argument is true, the `true-expression` is returned. Otherwise, the `false-expression` is returned. See the example for more information.
`std.atoi`	`INTEGER std.atoi(STRING s)`	Takes a string (which represents an integer) as an argument and returns its value. Example: `if (std.atoi(req.http.X-Decimal) == 42) { set req.http.X-TheAnswer = "Found"; }`

Function	Format	Description
`std.strstr`	`STRING std.strstr(STRING haystack, STRING needle)`	Finds the first occurrence of a byte string and returns its value. Example: `set req.http.X-qs = std.strstr(req.url, "?");`
`std.strtol`	`INTEGER std.strtol(STRING s, UINT base)`	Converts a string to an integer, using the second argument as base. Base can be `2` to `36`, or `0`. A `0` base means that base 10 (decimal) is used, unless the string has a `0x` or `0` prefix, in which case base 16 (hexadecimal) and base 8 (octal) are used respectively. For example, `std.strtol("0xa0", 0)` will return `160`. Example: `if (std.strtol(req.http.X-HexValue, 16) == 42) { set req.http.X-TheAnswer = "Found"; }`
`std.tolower`	`STRING std.tolower(STRING_LIST s)`	Changes the case of a string to lower case. For example, `std.tolower("HELLO");` will return `"hello"`. Example: `set beresp.http.x-nice = std.tolower("VerY");`

Function	Format	Description
`std.toupper`	`STRING std.toupper(STRING_LIST s)`	Changes the case of a string to upper case. For example, `std.toupper("hello");` will return `"HELLO"`. Example: `set beresp.http.x-scream = std.toupper("yes!");`
`std.str2ip`	`IP std.str2ip(STRING addr, STRING fallback)`	Converts the string address to an IP address (v4 or v6). For example, `if (std.str2ip("192.0.2.1", "192.0.2.2") ~ my_acl) {` where `192.0.2.2` is the fallback. If conversion fails, the fallback will be returned. Note that only the first result from DNS resolution is returned.
`std.ip`	`IP std.ip(STRING addr, STRING fallback)`	An alias of `std.str2ip`.
`std.ip2str`	`STRING std.ip2str(IP ip)`	Converts the IP address (v4 or v6) to a string. Example: `if (std.ip2str(std.str2ip("192.0.2.1", "192.0.2.2")) == "192.0.2.1")) {`
`std.strlen`	`INTEGER std.strlen(STRING s)`	Returns the length of the string. For example, `std.strlen("Hello world!");` will return `12` (because the string includes whitespaces and punctuation). Example: `if (std.strlen(req.http.Cookie > 1024)) { unset req.http.Cookie; }`

Function	Format	Description
`urlencode`	`STRING urlencode(STRING input)`	Encodes a string for use in a URL. This is also known as percent-encoding (https://en.wikipedia.org/wiki/Percent-encoding). For example, `urlencode("hello world");` will return `"hello%20world"`. Example: `set req.url = req.url "?cookie=" urlencode(req.http.Cookie);`
`urldecode`	`STRING urldecode(STRING input)`	Decodes a percent-encoded string. For example, `urldecode({"hello%20world+!"});` and `urldecode("hello%2025world+!");` will both return `"hello world !"` Example: `set req.http.X-Cookie = regsub(req.url, ".*\?cookie=", ""); set req.http.Cookie = urldecode(req.http.X-Cookie);`

Miscellaneous Features

Feature	Description
`goto`	Performs a one-way transfer of control to another line of code. See the example for more information.
`return`	Returns (with no return value) from a custom subroutine to exit early. See the example for more information.

Examples

Use the following examples to learn how to implement the features.

Goto

Similar to some programming languages, `goto` statements in VCL allow you perform a one-way transfer of control to another line of code. When using `goto`, jumps must always be forward, rather than to an earlier part of code.

This act of "jumping" allows you to do things like perform logical operations or set headers before returning lookup, error, or pass actions. These statements also make it easier to do things like jump to common error handling blocks before returning from a function. The `goto` statement works in custom subroutines.

```
sub vcl_recv {
  if (!req.http.Foo) {
    goto foo;
  }

foo:
  set req.http.Foo = "1";
}
```

Return

You can use `return` to exit early from a custom subroutine.

```
sub custom_subroutine {
  if (req.http.Cookie:user_id) {
    return;
  }

  # do a bunch of other stuff
}
```

If

You can use `if()` as a construct to make simple conditional expressions more concise.

```
set req.http.foo-status = if(req.http.foo, "present", "absent");
```

§ Mixing and matching Fastly VCL with custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl)

❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Fastly Varnish syntax is specifically compatible with Varnish 2.1.5 (https://www.varnish-cache.org/docs/2.1). We run a custom version with added functionality and our VCL parser has its own pre-processor. To mix and match Fastly VCL with your custom VCL successfully, remember the following:

- **You can only restart Varnish requests three times.** This limit exists to prevent infinite loops.
- **VCL doesn't take kindly to Windows newlines (line breaks).** It's best to avoid them entirely.
- **It's best to use `curl -X PURGE` to initiate purges via API (/api/purge).** To restrict access to purging, check for the `FASTLYPURGE` method not the `PURGE` method. When you send a request to Varnish to initiate a purge, the HTTP method that you use is "PURGE", but it has already been changed to "FASTLYPURGE" by the time your VCL runs that request.
- **If you override TTLs with custom VCL, your default TTL set in the configuration (/guides/performance-tuning/serving-stale-content) will not be honored** and the expected behavior may change.

Inserting custom VCL in Fastly's VCL boilerplate

⚠ **DANGER:** Include all of the Fastly VCL boilerplate as a template in your custom VCL file, especially the VCL macro lines (they start with `#FASTLY`). VCL macros expand the code into generated VCL. Add your custom code *in between* the different sections as shown in the example unless you specifically intend to override the VCL at that point.

Custom VCL placement example

```
sub vcl_miss {
  # my custom code
  if (req.http.User-Agent ~ "Googlebot") {
    set req.backend = F_special_google_backend;
  }
#FASTLY miss
  return(fetch);
}
```

Fastly's VCL boilerplate

```
sub vcl_recv {
#FASTLY recv

  if (req.request != "HEAD" && req.request != "GET" && req.request != "FASTLYPURGE") {
    return(pass);
  }

  return(lookup);
}

sub vcl_fetch {
#FASTLY fetch

  if ((beresp.status == 500 || beresp.status == 503) && req.restarts < 1 && (req.request == "GE
T" || req.request == "HEAD")) {
    restart;
  }

  if (req.restarts > 0) {
    set beresp.http.Fastly-Restarts = req.restarts;
  }

  if (beresp.http.Set-Cookie) {
    set req.http.Fastly-Cachetype = "SETCOOKIE";
    return(pass);
  }

  if (beresp.http.Cache-Control ~ "private") {
    set req.http.Fastly-Cachetype = "PRIVATE";
    return(pass);
  }

  if (beresp.status == 500 || beresp.status == 503) {
    set req.http.Fastly-Cachetype = "ERROR";
    set beresp.ttl = 1s;
    set beresp.grace = 5s;
    return(deliver);
  }

  if (beresp.http.Expires || beresp.http.Surrogate-Control ~ "max-age" || beresp.http.Cache-Con
trol ~ "(s-maxage|max-age)") {
    # keep the ttl here
  } else {
    # apply the default ttl
    set beresp.ttl = 3600s;
  }

  return(deliver);
}

sub vcl_hit {
```

```
#FASTLY hit

  if (!obj.cacheable) {
    return(pass);
  }
  return(deliver);
}

sub vcl_miss {
#FASTLY miss
  return(fetch);
}

sub vcl_deliver {
#FASTLY deliver
  return(deliver);
}

sub vcl_error {
#FASTLY error
}

sub vcl_pass {
#FASTLY pass
}

sub vcl_log {
#FASTLY log
}
```

§ Overriding which IP address the geolocation features use (/guides/vcl/overriding-which-ip-address-the-geolocation-features-use)

By default geolocation (/guides/vcl/geolocation-related-vcl-features) lookup is based on the IP address of the user. In some cases, such as with traffic through proxies, this type of lookup doesn't work properly. In particular, users of Opera Mini always browse through a proxy and the true IP address appears in the X-Forwarded-For header (https://dev.opera.com/articles/opera-mini-request-headers/#x-forwarded-for). Similarly, the Amazon Silk browser (https://developer.amazon.com/appsandservices/community/post/TxOMW3RNF3FYRK/Amazon-Silk-Tips-for-Site-Owners) can optionally come through a proxy, indicated via the User Agent

string. In cases like these, the X-Forwarded-For header will contain a comma-separated list of IPs instead of just one IP. Attempting a geolocation lookup on anything other than a single IP will result in a lookup failure, so you need to ensure the lookup is done on the end-user's IP only.

To work around this and to account for both the Opera Mini and Amazon Silk browsers, you would use code like this in `vcl_recv`:

```
if ((req.http.X-OperaMini-Features || req.http.User-Agent ~ " Silk-Accelerated=true$") && req.h
ttp.X-Forwarded-For ){
  set client.geo.ip_override = regsub(req.http.X-Forwarded-For, ",.+$", "");
}
```

which tells Fastly to use only the first IP in the X-Forwarded-For header as the value for the IP address. If it is not available, then the code will fall back to using the IP address of the client.

Finally, just in case there's some scenario or browser (/guides/debugging/browser-recommendations-when-using-the-fastly-web-interface) we haven't anticipated, you can also override based on an arbitrary header:

```
set client.geo.ip_override = req.http.Custom-IP-Override;
```

Setting this variable will force the geolocation information to be reloaded.

❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

---

§ Previewing and testing VCL before activating it (/guides/vcl/previewing-and-testing-vcl-before-activating-it)

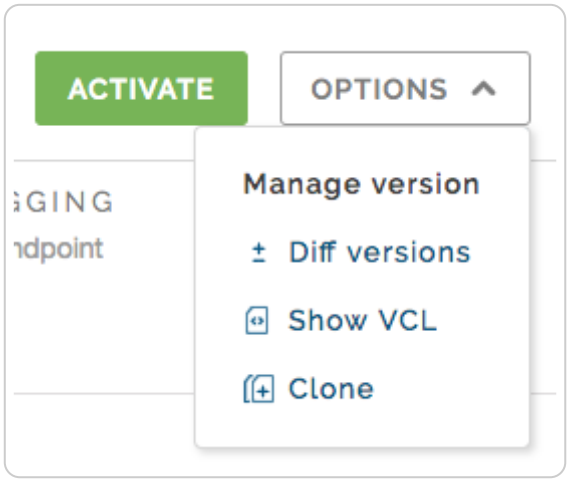
Any time you upload VCL files (/guides/vcl/uploading-custom-vcl) you can preview and test the VCL prior to activating a new version of your service.

Previewing VCL before activation

To preview VCL prior to activating a service version.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.

3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Options** button to open the **Manage version** menu and select **Show VCL**.



The VCL preview page appears.

Testing VCL configurations

You don't need a second account to test your VCL configurations. We recommend adding a new service within your existing account that's specifically designed for testing. A name like "QA" or "testing" or "staging" makes distinguishing between services much easier.

Once created, simply point your testing service to your testing or QA environment. Edit your Fastly configurations for the testing service as if you were creating them for production. Preview your VCL, test things out, and tweak them to get them perfect.

When your testing is complete, make the same changes in your production service that you made to your testing service. If you are using custom VCL, upload the VCL file (/guides/vcl/uploading-custom-vcl) to the production service you'll be using.

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

§ Query string manipulation VCL features (/guides/vcl/query-string-manipulation-vcl-features)

Fastly provides a number of extensions to VCL (/guides/vcl/guide-to-vcl#fastlys-vcl-extensions), including several functions for query-string manipulation based on Dridi Boukelmoune's vmod-querystring (https://github.com/Dridi/libvmod-querystring) for Varnish.

Functions

Function	Description	Example
`querystring.add(<string>, <string>, <string>)`	Returns the given URL with the given parameter name and value appended to the end of the query string. The parameter name and value will be URL-encoded when added to the query string.	`set req.url = querystring.add(req.url, "foo", "bar");`

Function	Description	Example
`querystring.clean(<string>)`	Returns the given URL without empty parameters. The query-string is removed if empty (either before or after the removal of empty parameters). Note that a parameter with an empty value does not constitute an empty parameter, so a query string "?something" would not be cleaned.	`set req.url = querystring.clean(req.url);`
`querystring.filter(<string>, <string_list>)`	Returns the given URL without the listed parameters.	`set req.url = querystring.filter(req.url, "utm_source" + querystring.filte + "utm_medium" + querystring.filtersep() + "utm_campaign");`
`querystring.filter_except(<string>, <string_list>)`	Returns the given URL but only keeps the listed parameters.	`set req.url = querystring.filter_except(req.ur + querystring.filtersep() + "p");`

Function	Description	Example
`querystring.filtersep()`	Returns the separator needed by the `filter` and `filter_except` functions.	
`querystring.globfilter(<string>, <string>)`	Returns the given URL without the parameters matching a glob.	`set req.url = querystring.globfilter(req.url, "utm_*");`
`querystring.globfilter_except(<string>, <string>)`	Returns the given URL but only keeps the parameters matching a glob.	`set req.url = querystring.globfilter_except(req "sess*");`
`querystring.regfilter(<string>, <string>)`	Returns the given URL without the parameters matching a regular expression.	`set req.url = querystring.regfilter(req.url, "^utm_.*");`
`querystring.regfilter_except(<string>, <string>)`	Returns the given URL but only keeps the parameters matching a regular expression.	`set req.url = querystring.regfilter_except(req "^(q|p)$");`
`querystring.remove(<string>)`	Returns the given URL with its query-string removed.	`set req.url = querystring.remove(req.url);`

Function	Description	Example
`querystring.set(<string>, <string>, <string>)`	Returns the given URL with the given parameter name set to the given value, replacing the original value and removing any duplicates. If the parameter is not present in the query string, the parameter will be appended with the given value to the end of the query string. The parameter name and value will be URL-encoded when set in the query string.	`set req.url = querystring.set(req.url, "foo", "baz");`
`querystring.sort(<string>)`	Returns the given URL with its query-string sorted.	`set req.url = querystring.sort(req.url);`

Examples

In your VCL, you could use `querystring.regfilter_except` as follows:

```
import querystring;

sub vcl_recv {
    # return this URL with only the parameters that match this regular expression
    set req.url = querystring.regfilter_except(req.url, "^(q|p)$");
}
```

You can use `querystring.regfilter` to specify a list of arguments that must not be removed (everything else will be) with a negative look-ahead expression:

```
set req.url = querystring.regfilter(req.url, "^(?!param1|param2)");
```

§ Randomness-related VCL features (/guides/vcl/randomness-related-vcl-features)

Fastly provides several functions in VCL (/guides/vcl/) that control randomness-related activities.

⚠ **WARNING:** We use BSD random number functions from the GNU C Library (http://www.gnu.org/software/libc/manual/html_node/BSD-Random.html), not true randomizing sources. These VCL functions should not be used for cryptographic (/guides/vcl/cryptographic-and-hashing-related-vcl-functions) or security purposes.

Random strings

Use the function `randomstr(length [, characters])`. When characters aren't provided, the default will be the 64 characters of `A-Za-z0-9_-`.

```
sub vcl_deliver {
  set resp.http.Foo = "randomstuff=" randomstr(10);
  set resp.http.Bar = "morsecode=" randomstr(50, ".-"); # 50 dots and dashes
}
```

Random content cookies in pure VCL

```
sub vcl_deliver {
  add resp.http.Set-Cookie = "somerandomstuff=" randomstr(10) "; expires=" now + 180d "; path
=/;";
}
```

This adds a cookie named "somerandomstuff" with 10 random characters as value, expiring 180 days from now.

Random decisions

Use the function `randombool( _numerator_, _denominator_ )`, which has a numerator/denominator chance of returning true.

```
sub vcl_recv {
  if (randombool(1, 4)) {
    set req.http.X-AB = "A";
  } else {
    set req.http.X-AB = "B";
  }
}
```

This will add a X-AB header to the request, with a 25% (1 out of 4) chance of having the value "A", and 75% chance of having the value "B".

The `randombool()` function accepts INT function return values, so you could do something this:

```
if (randombool(std.atoi(req.http.Some-Header), 100))  {
  # do something
}
```

Another function, `randombool_seeded()`, takes an additional seed argument. Results for a given seed will always be the same. For instance, in this example the value of the response header will always be `no`:

```
if (randombool_seeded(50, 100, 12345)) {
  set resp.http.Seeded-Value = "yes";
} else {
  set resp.http.Seeded-Value = "no";
}
```

This could be useful for stickiness. For example, if you based the seed off of something that identified a user, you could perform A/B testing without setting a special cookie.

⚠ **WARNING:** The `randombool` and `randombool_seeded` functions do not use secure random numbers and should not be used for cryptographic purposes.

---

§ Response Cookie handling (/guides/vcl/response-cookie-handling)

The traditional way to read response cookies in VCL is to inspect either the `beresp.http.Set-Cookie` or the `resp.http.Set-Cookie` variables and then extract values using regular expressions. However this is not ideal since attempting to parse potentially complicated or quoted strings with regular expressions is brittle and prone to being tripped up by edge cases. It also doesn't allow for reading multiple headers with the same name such as when an origin sends multiple `Set-Cookie` headers. Because of these two reasons Fastly supports a method for extracting a named value out of `Set-Cookie` headers no matter how many there are.

To access a named value simply use the function with either `beresp` or `resp` depending on what part of the request you're in - so either

```
setcookie.get_value_by_name(beresp, "name")
```

or

```
setcookie.get_value_by_name(resp, "name")
```

as appropriate, replacing `"name"` with whatever the name of the value is. So for example, given this HTTP response from an origin

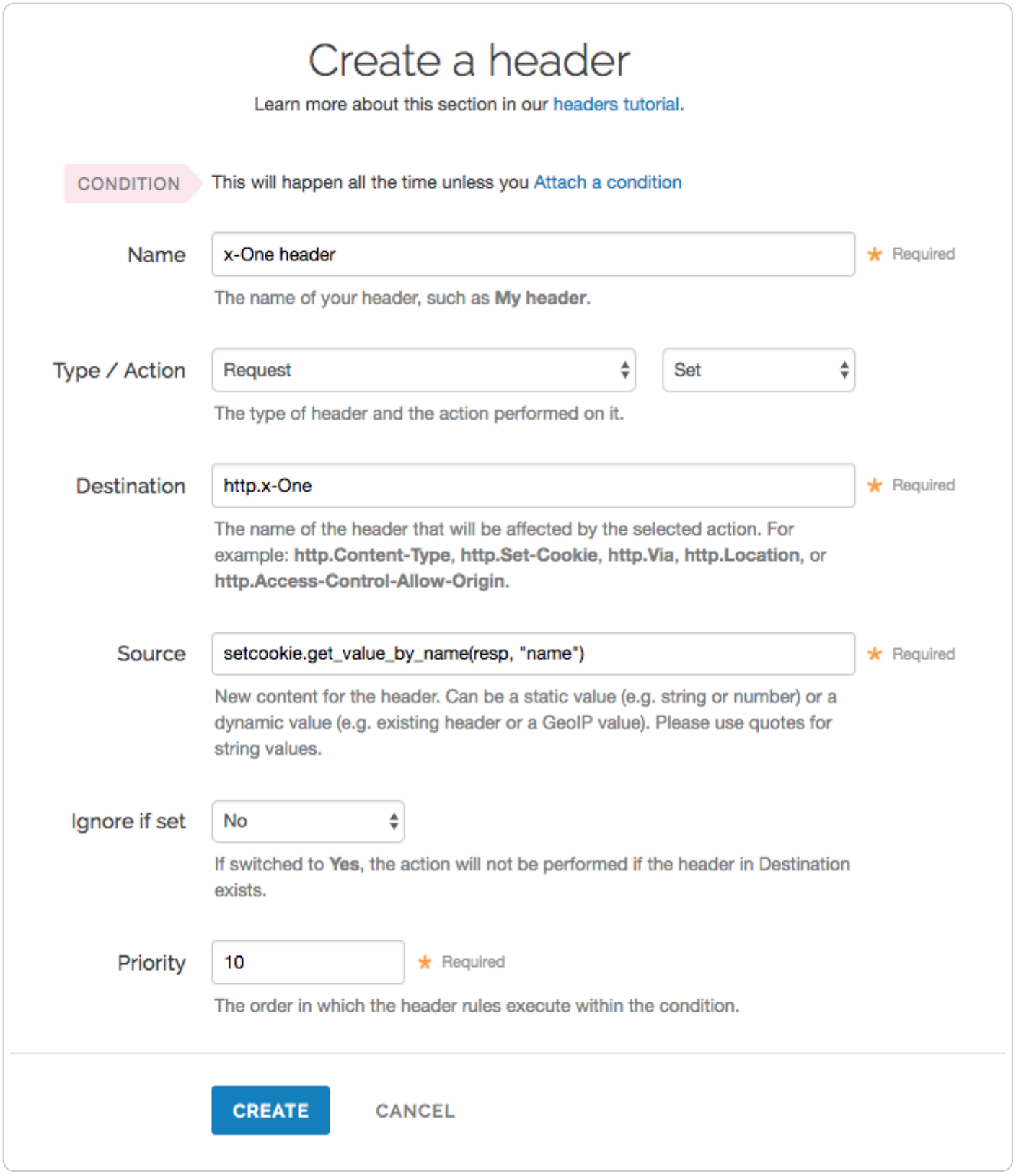
```
HTTP/1.1 200 OK
Cache-Control: max-age=60
Content-Type: text/html; charset=utf-8
Content-Length: 80806
Accept-Ranges: bytes
Date: Tue, 11 Aug 2015 19:00:04 GMT
Age: 123
Connection: keep-alive
Set-Cookie: one=a; httponly; secure
Set-Cookie: two=b or not to b; httponly
```

then using the function like this

```
set resp.http.X-One = setcookie.get_value_by_name(resp, "one");
set resp.http.X-Two = setcookie.get_value_by_name(resp, "two");
```

will set `resp.http.X-One` to be "a" and `resp.http.X-Two` to "b or not to b".

This logic can be used in uploaded custom VCL (/guides/vcl/uploading-custom-vcl), as well as throughout the web interface. For example:



§ Setting up redundant origin servers (/guides/vcl/setting-up-redundant-origin-servers)

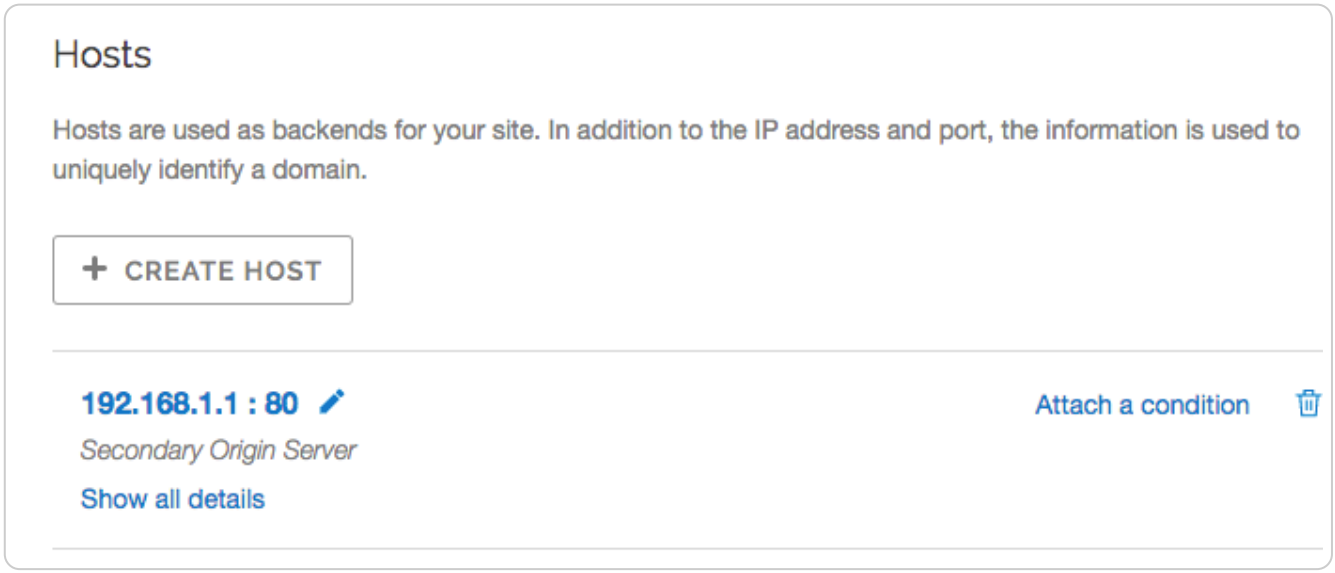
Sometimes you want to set up two different origin servers, one as a primary and one as a backup in case the primary becomes unavailable. You can do this via the web interface or using custom VCL.

> **NOTE:** Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com (mailto:sales@fastly.com) to enable more than five origin servers per service in your account.

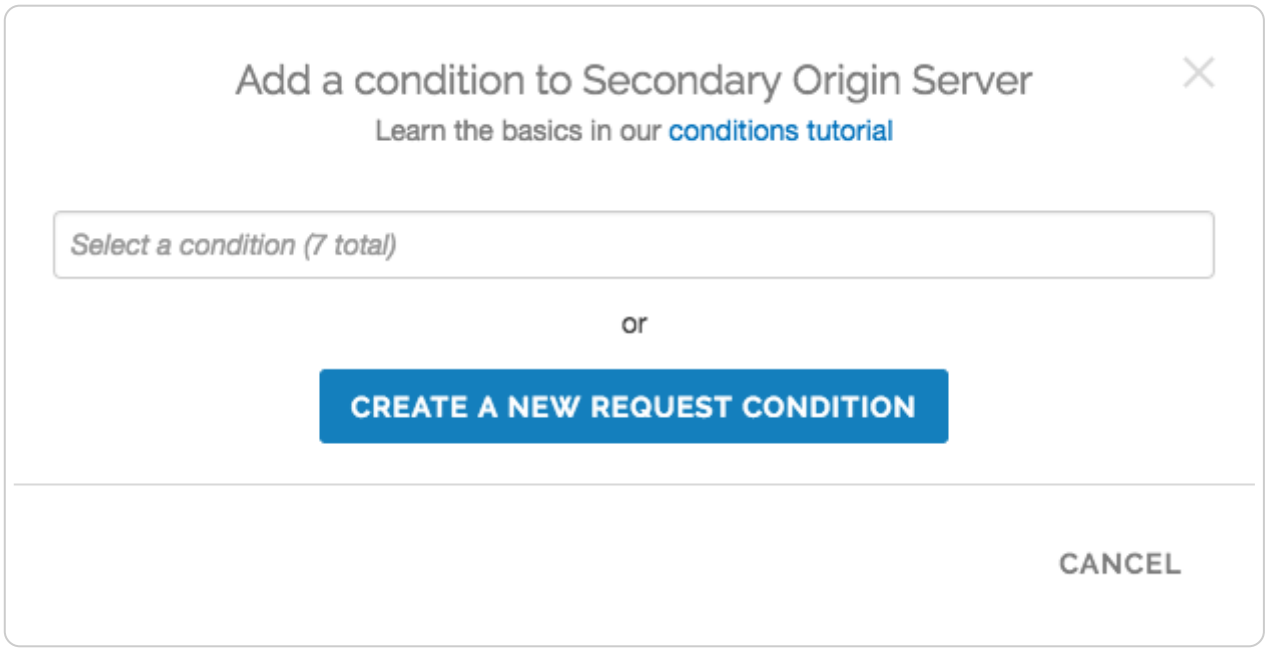
Using the user interface

Set up redundant origins via the user interface using these steps.

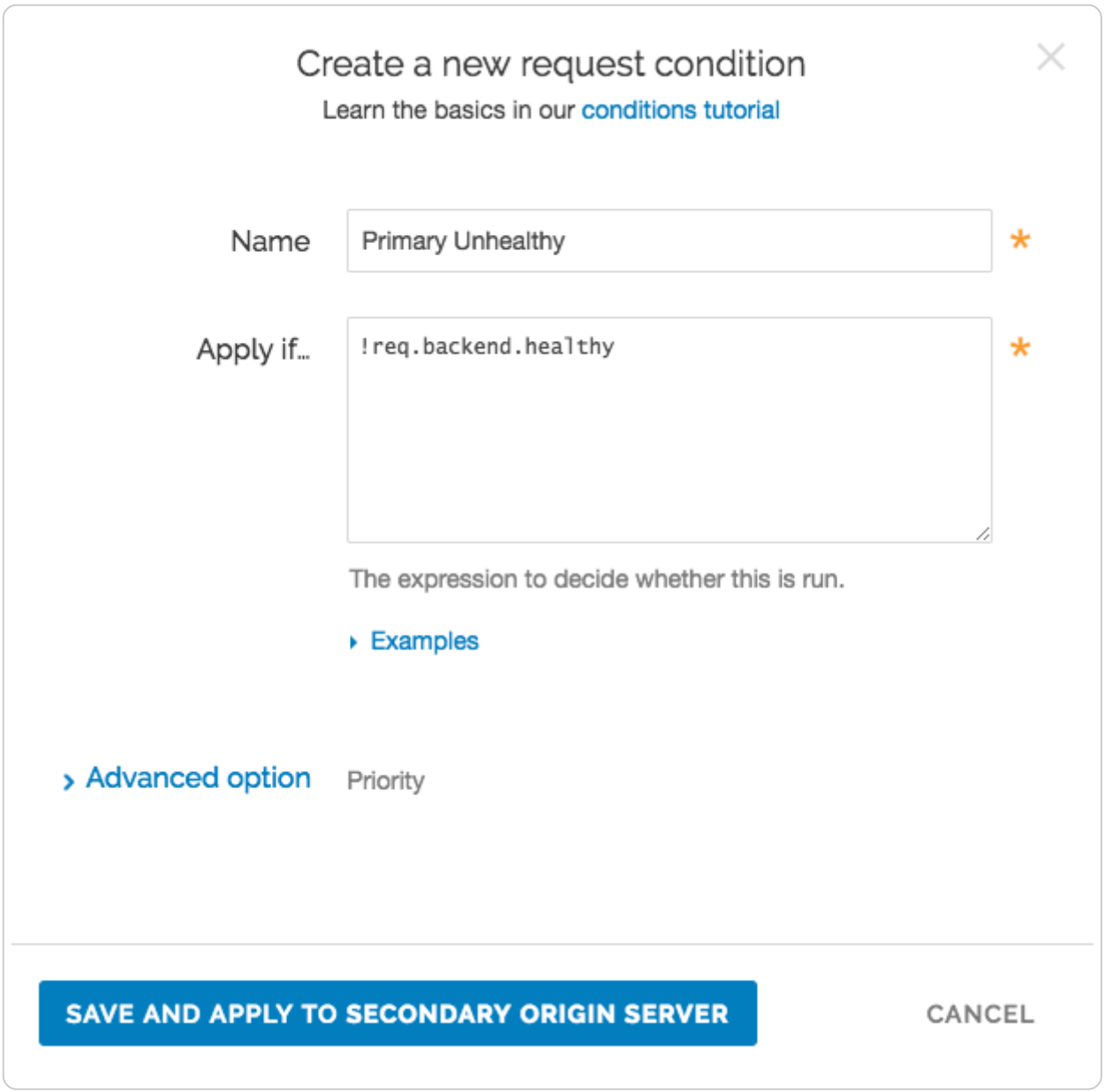
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. In the **Health Checks** area, define a health check (/guides/basic-configuration/health-checks-tutorial) and assign it to the primary origin server.
6. In the **Hosts** area, find your secondary origin server and click the **Attach a condition** link.



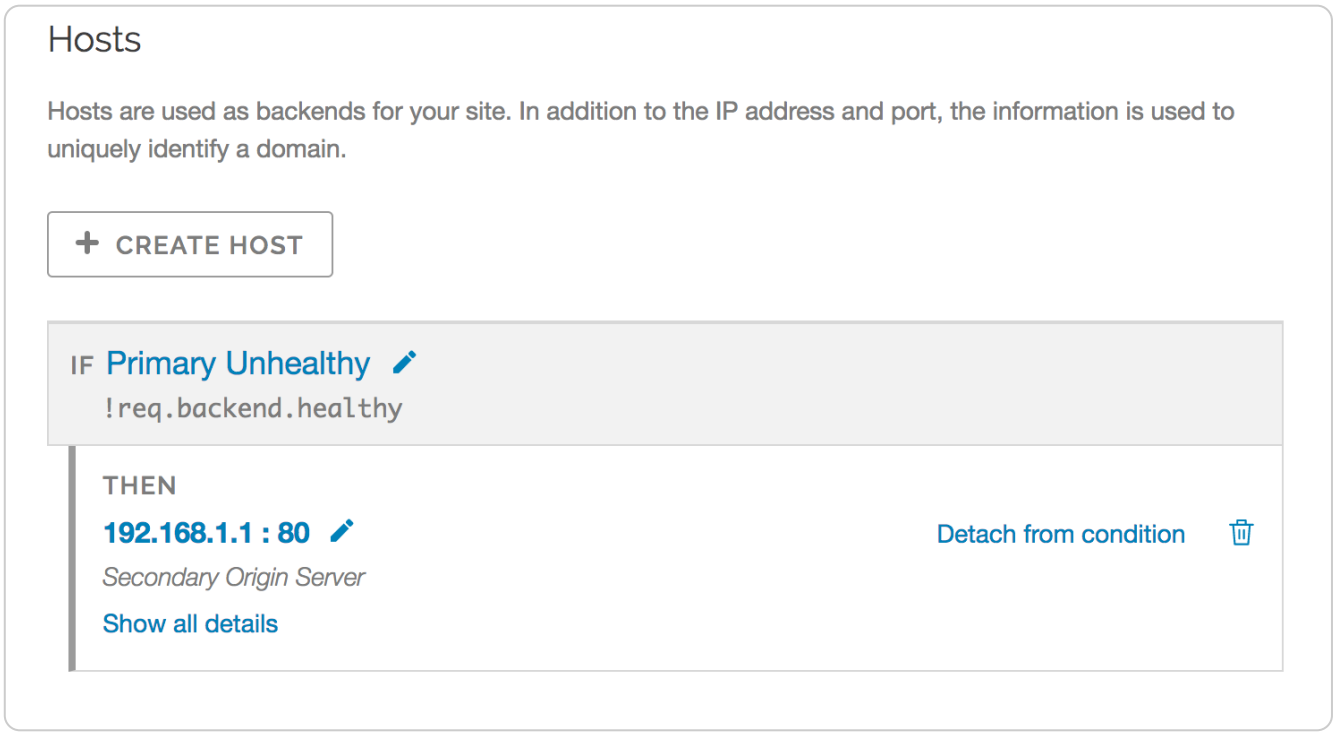
The Add a condition window appears.



7. Click **Create a new request condition**. The Create a new request condition window appears.



8. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type the name of your request condition (for example, `Primary Unhealthy`).
   - In the **Apply if** field, type `!req.backend.healthy`.
9. Click the **Save and apply to** button. The Hosts area now displays the condition that must be met (Primary Unhealthy) in order for your secondary origin server to begin being used.



Once you've added the condition to your secondary origin server, the VCL generated by Fastly will reflect the new condition.

10. Preview the VCL (/guides/vcl/previewing-and-testing-vcl-before-activating-it), and confirm the following snippets appear in `vcl_recv`:

```
# default conditions
set req.backend = F_primary;
```

```
# Request Condition: primary unhealthy Prio: 10
if (!req.backend.healthy) {
  set req.backend = F_secondary;
}
#end condition
```

Using custom VCL

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Set up redundant origins with custom VCL using these steps.

1. In the Fastly web interface, define a health check (/guides/basic-configuration/health-checks-tutorial) and assign it to the primary origin server.

2. Copy the boilerplate VCL from our guide on mixing Fastly VCL with custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl), and paste it into a new file.
3. Replace the `vcl_recv` sub with:

   ```
   sub vcl_recv {
   #FASTLY recv
     set req.backend = F_<primary_origin>;
     if (!req.backend.healthy) {
       set req.backend = F_<secondary_origin>;
     }
     if (req.request != "HEAD" && req.request != "GET" && req.request != "FASTLYPURGE") {
       return(pass);
     }
     return(lookup);
   }
   ```

   To find the exact backend names, view the generated VCL (/guides/vcl/previewing-and-testing-vcl-before-activating-it).
4. Upload (/guides/vcl/uploading-custom-vcl) your VCL file.

---

§ Size-related VCL variables (/guides/vcl/size-related-vcl-variables)

To allow better reporting, Fastly has added several variables (/guides/vcl/guide-to-vcl#fastlys-vcl-extensions) to VCL to give more insight into what happened in a request.

Name	Where	Description
`req.bytes_read`	deliver log	How big a request was in total bytes.
`req.header_bytes_read`	all	How big the header of a request was in total bytes.
`req.body_bytes_read`	deliver log	How big the body of a request was in total bytes.
`resp.bytes_written`	log	How many bytes in total were sent as a response.
`resp.header_bytes_written`	log	How many bytes were written for the header of a response.
`resp.body_bytes_written`	log	How many bytes were written for body of a response.
`resp.completed`	log	Whether the response completed successfully or not.

§ Support for the Edge-Control header (/guides/vcl/support-for-the-edge-control-header)

VCL provides the building blocks to access information inside the Edge-Control response header field from the origin. We support this by honoring `cache-maxage` from Edge-Control as the time to live (TTL) of the object on the Fastly edge, and honoring `downstream-ttl` from Edge-Control as the TTL to be sent down from the Fastly edge to the end user's browser.

In order to incorporate this Edge-Control header support, include the following snippet in your `vcl_fetch` function via custom VCL (/guides/vcl/uploading-custom-vcl):

```
if (parse_time_delta(subfield(beresp.http.Edge-Control, "downstream-ttl")) >= 0) {
  set beresp.http.Cache-Control = "max-age=" parse_time_delta(subfield(beresp.http.Edge-Control
, "downstream-ttl"));
}

if (parse_time_delta(subfield(beresp.http.Edge-Control, "cache-maxage")) >= 0) {
  set beresp.ttl = parse_time_delta(subfield(beresp.http.Edge-Control, "cache-maxage"));
}
```

The subfield function parses the Edge-Control field for subfields, and the `parse_time_delta` function converts time values like "7m" into a number of seconds. You can then use that number of seconds to populate `beresp.ttl` (the TTL of the object on the Fastly edge) or you can use it to construct a Cache-Control header field for downstream. The `parse_time_delta` function will return -1 if the subfield is not well-formed as a time value, or if it is entirely absent. The above snippet honors `cache-maxage` and `downstream-ttl` from Edge-Control if present and usable.

> ❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

§ TLS and HTTP/2 VCL variables and functions (/guides/vcl/tls-and-http2-vcl-variables-and-functions)

Fastly has added several variables (/guides/vcl/guide-to-vcl#fastlys-vcl-extensions) that expose information about the TLS and HTTP/2 attributes of a request. When using these variables, remember the following:

- These variables are currently only allowed to appear within the VCL hooks `vcl_recv`, `vcl_hash`, `vcl_deliver` and `vcl_log`.
- Requests made with HTTP/2 will appear in custom logs (/guides/streaming-logs/custom-log-formats) as HTTP1.1 because those requests will already have been decrypted by the time Varnish sees it. Specifically, the `%r` variable will not accurately represent the type of HTTPX request being processed.

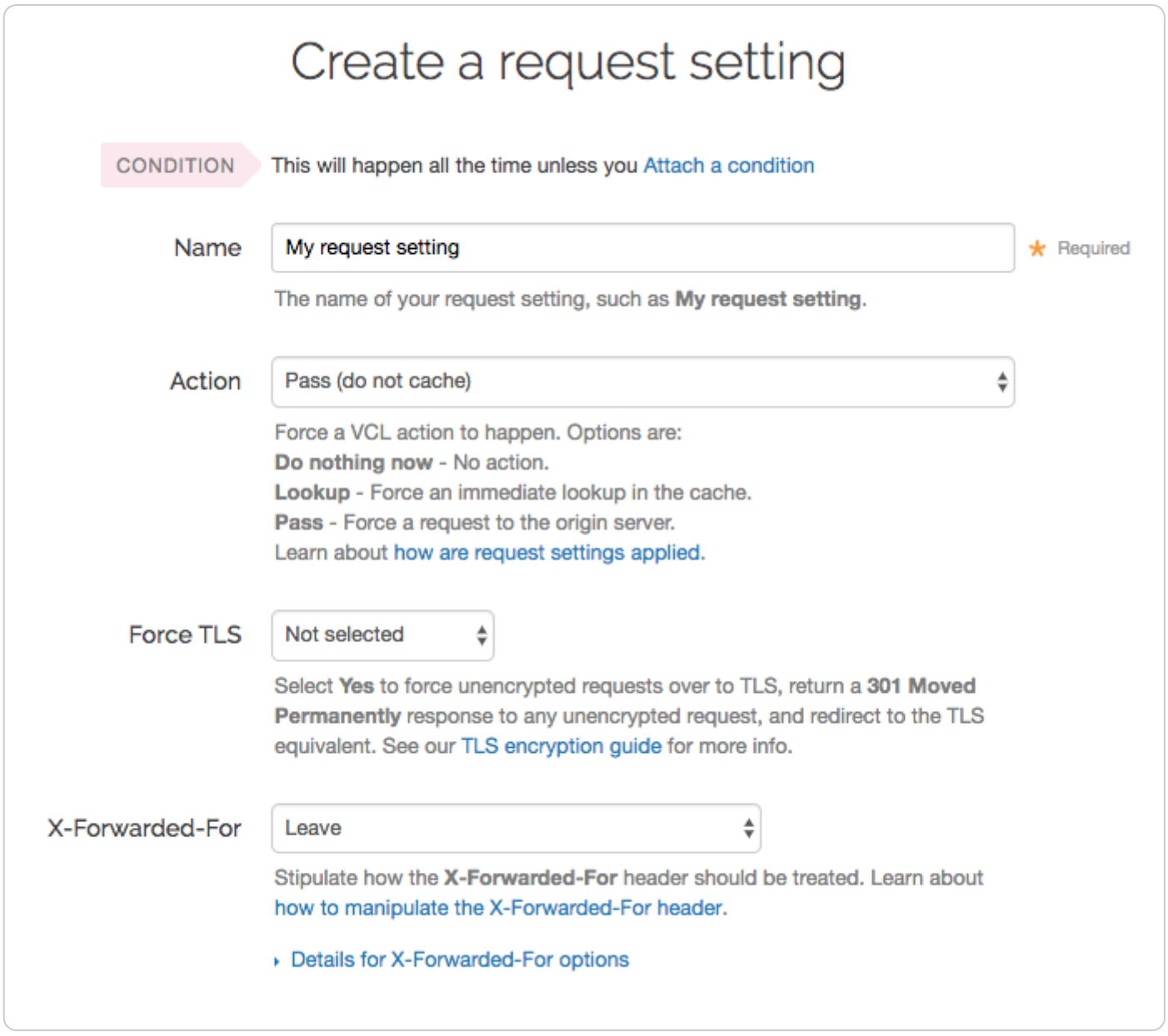
Feature	Description
`fastly_info.is_h2`	Whether or not the request was made using HTTP/2. Returns a boolean value.
`fastly_info.h2.is_push`	If the request was made over HTTP/2, whether or not the request is part of a push request. Returns a boolean value.
`fastly_info.h2.stream_id`	If the request was made over HTTP/2, the underlying HTTP/2 stream ID.
`h2.push(<string>)`	Triggers an HTTP/2 server push of the asset passed into the function as the input-string.
`tls.client.protocol`	The TLS protocol version this connection is speaking over. Example: `"TLSv1.2"`
`tls.client.servername`	The Server Name Indication (SNI) the client sent in the `ClientHello` TLS record. Returns `""` if the client did not send SNI. Returns `NULL` (the undefined string) if the request is not a TLS request.
`tls.client.cipher`	The cipher suite used to secure the client TLS connection. Example: `"ECDHE-RSA-AES128-GCM-SHA256"`
`tls.client.ciphers_sha`	A SHA1 of the cipher suite identifiers sent from the client as part of the TLS handshake, represented in base64.
`tls.client.tlsexts_sha`	A SHA1 of the TLS extension identifiers sent from the client as part of the TLS handshake, represented in base64.

§ Understanding the different PASS action behaviors (/guides/vcl/understanding-the-

different-pass-action-behaviors)

Passing with a request setting and with a cache setting triggers very different behavior in Varnish (/guides/vcl/guide-to-vcl). Within VCL, passing with a request setting is the same as `return(pass)` in `vcl_recv`. Passing with a cache setting is the same as `return(pass)` in `vcl_fetch`. If you are familiar with Varnish 3+, passing with a cache setting is equivalent to `return(hit_for_pass)`.

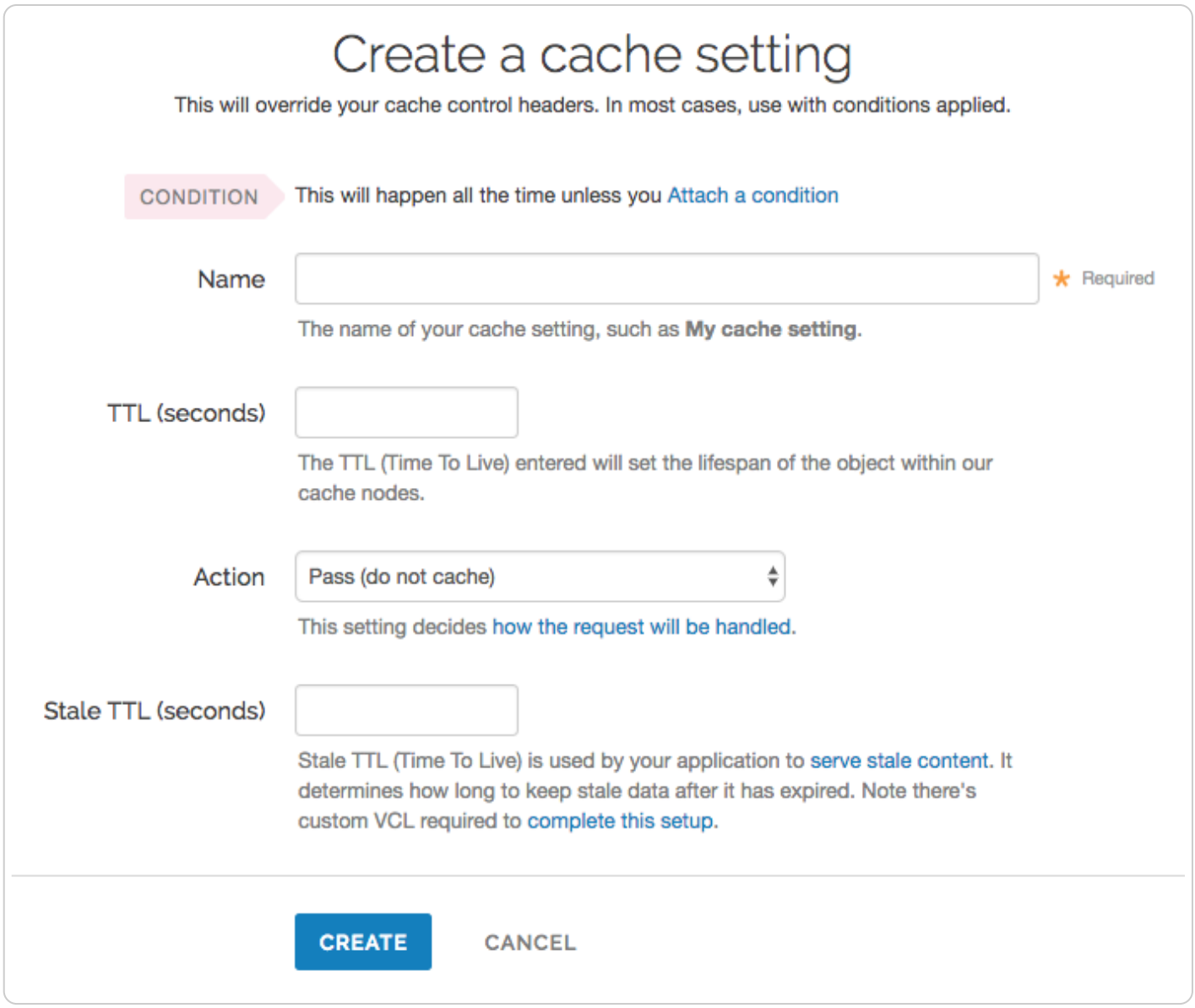
Using a request setting



Passing with a request setting translates within your generated VCL to `return(pass)` in `vcl_recv`. Varnish will not perform a lookup to see if an object is in cache and the response from the origin will not be cached.

Passing in this manner disables request collapsing. Normally simultaneous requests for the same object that result in cache misses will be collapsed down to a single request to the origin. While the first request is sent to the origin, the other requests for that object are queued until a response is received. When requests are passed in `vcl_recv`, they will all go to the origin separately without being collapsed.

Using a cache setting



Passing with a cache setting translates within your generated VCL to `return(pass)` in `vcl_fetch`. At this point in the flow of a request, Varnish has performed a lookup and determined that the object is not in cache. A request to the origin has been made; however, in vcl_fetch we have determined that the response is not cacheable. In Fastly's default VCL, this can happen based on the presence of a `Set-Cookie` response header from the origin.

Passing in `vcl_fetch` is often not desirable because request collapsing is *not* disabled. This makes sense since Varnish is not aware in `vcl_recv` that the object is uncacheable. On the first request for an object that will be later passed in `vcl_fetch`, all other simultaneous cache misses will be queued. Once the response from the origin is received and Varnish has realized that the request should be passed, the queued requests are sent to the origin.

This creates a scenario where two users request an object at the same time, and one user must wait for the other before being served. If these requests were passed in `vcl_recv`, neither user would need to wait.

To get around this disadvantage, when a request is passed in `vcl_fetch`, Varnish creates what is called a hit-for-pass object. These objects have their own TTLs and while they exist, Varnish will pass any requests for them as if the pass had been triggered in `vcl_recv`. For this reason, it is important to set a TTL that makes sense for your case when you pass in `vcl_fetch`. All future requests for the object will be passed until the hit-for-pass object expires. Hit-for-pass objects can also be purged like any other object.

Even with this feature, there will be cases where simultaneous requests will be queued and users will wait. Whenever there is not a hit-for-pass object in cache, these requests will be treated as if they are normal cache misses and request collapsing will be enabled. Whenever possible it is best avoid relying on passing in `vcl_fetch`.

Using req.hash_always_miss and req.hash_ignore_busy

Setting `req.hash_always_miss` forces a request to miss whether it is in cache or not. It is different when passing in `vcl_recv` in that the response will be cached and request collapsing will not be disabled. Later on the request can still be passed in `vcl_fetch` if desired.

A second relevant variable is `req.hash_ignore_busy`. Setting this to true disables request collapsing so that each request is sent separately to origin. When `req.hash_ignore_busy` is enabled all responses will be cached and each response received from the origin will overwrite the last. Future requests for the object that are served from cache will receive the copy of the object from the last cache miss to complete. `req.hash_ignore_busy` is used mostly for avoiding deadlocks in complex multi-Varnish setups.

Setting both these variables can be useful to force requests to be sent separately to the origin while still caching the responses.

§ Uploading custom VCL (/guides/vcl/uploading-custom-vcl)

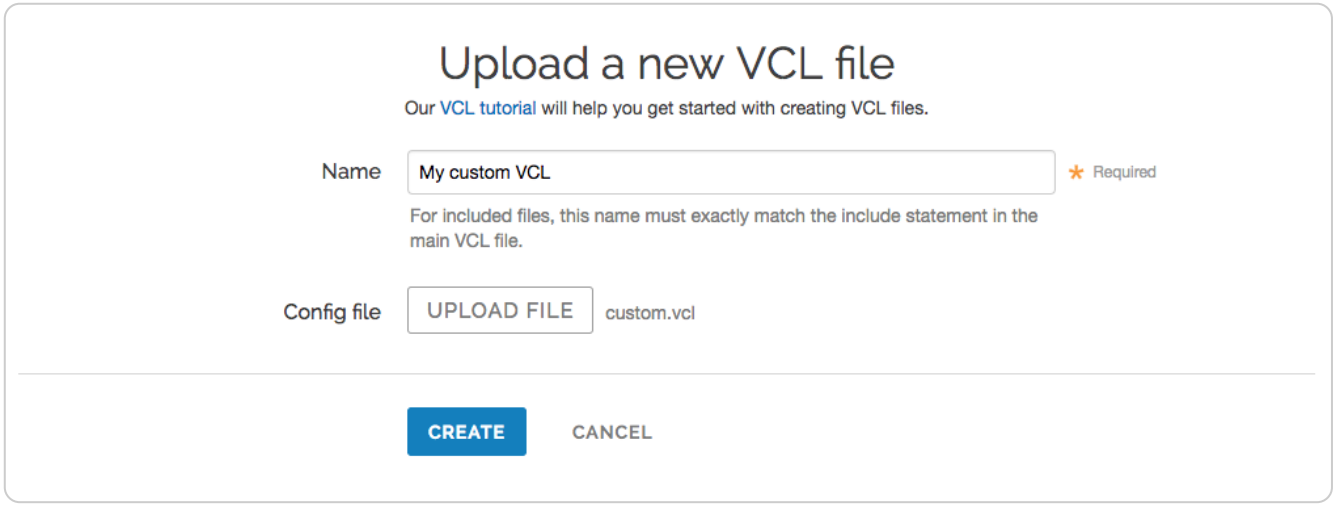
Fastly allows you create your own Varnish Configuration Language (VCL) files with specialized configurations. By uploading custom VCL files, you can use custom VCL and Fastly VCL together at the same time (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl). Keep in mind that your custom VCL always takes precedence over VCL generated by Fastly.

❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Uploading a VCL file

After Fastly support has enabled the ability to upload custom VCL to your account, follow these instructions to upload a custom VCL file:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Custom VCL** tab. The Custom VCL page appears.
5. Click the **Upload a new VCL file** button. The Upload a new VCL file page appears.

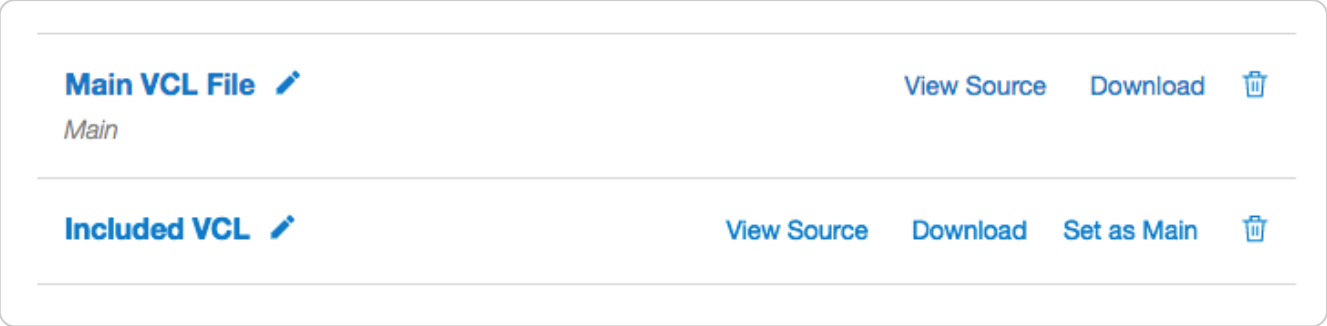


6. In the **Name** field, enter the name of the VCL file. For included files, this name must match the include statement in the main VCL file. See how to include additional VCL configurations for more information.

7. Click **Upload file** and select a file to upload. The name of the uploaded file appears next to the button.

   > ❗ **IMPORTANT:** Don't upload generated VCL that you've downloaded from the Fastly web interface. Instead, edit and then upload a copy of Fastly's VCL boilerplate (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl#fastlys-vcl-boilerplate) to avoid errors.

8. Click the **Create** button. The VCL file appears in the Varnish Configurations area.

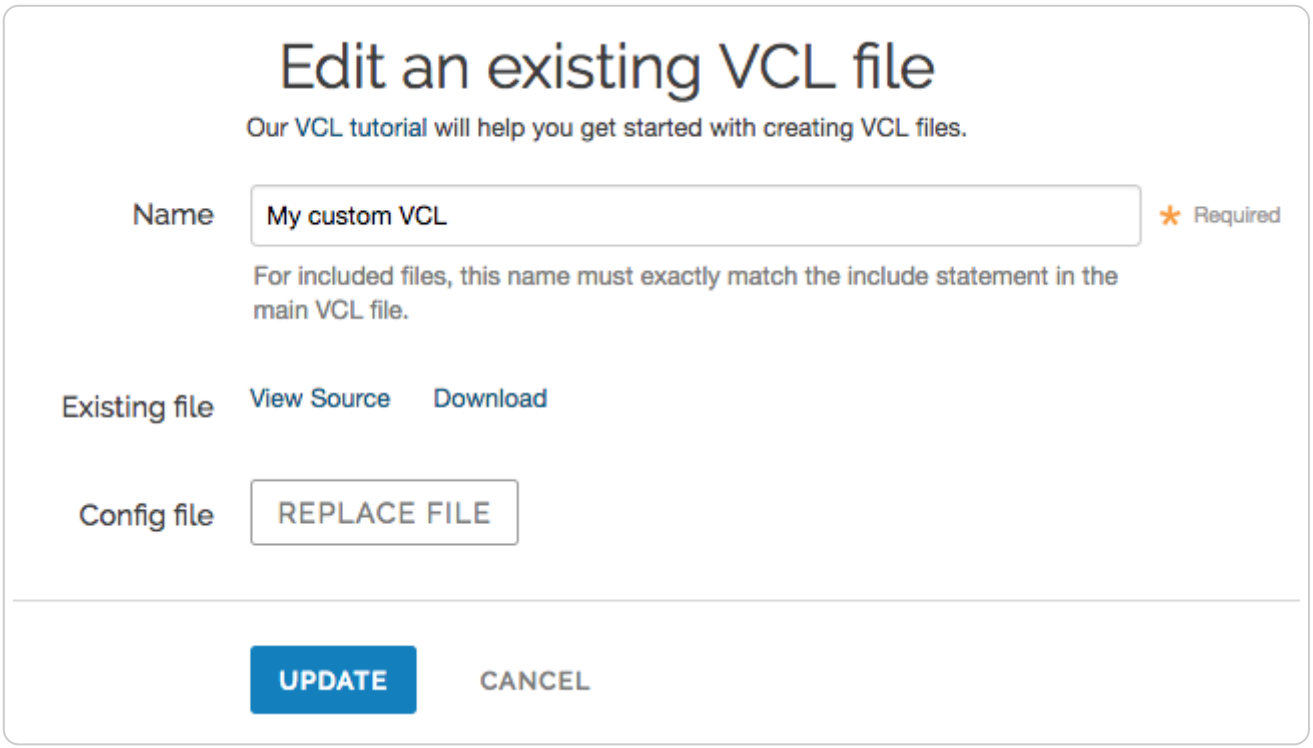


9. Click the **Activate** button to deploy your configuration changes.

Editing a VCL file

To edit an existing VCL file, follow these instructions:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Custom VCL** tab. The Custom VCL page appears.
5. In the **Varnish Configurations** area, click the VCL file you want to edit. The Edit an existing VCL file page appears.



6. In the **Name** field, optionally enter a new name of the VCL file.
7. Click the **Download** link to download the appropriate file.
8. Make the necessary changes to your file and save them.
9. Click the **Replace file** button and select the file you updated. The selected file replaces the current VCL file and the file name appears next to the button.
10. Click the **Update** button to update the VCL file in the Fastly application.
11. Click the **Activate** button to deploy your configuration changes.

Including additional VCL configurations

You can apply additional VCL files along with your main VCL by including their file names in the main VCL file using the syntax `include "VCL Name"` where `VCL Name` is the name of an included VCL object you've created.

For example, if you've created an included VCL object called "ACL" (to use an access control list (/guides/vcl/using-access-control-lists) for code manageability) and the file is named `acl.vcl`, your main VCL configuration file would need to contain this line:

```
include "ACL"
```

---

§ VCL regular expression cheat sheet (/guides/vcl/vcl-regular-expression-cheat-

sheet)

★ **TIP:** Varnish (/guides/vcl/) regular expressions are case sensitive and forward slashes don't need to be escaped. To disable case sensitivity, add `(?i)` to the start of your expression.

Basic matching

```
req.url == "/phrase"
```

Matches only if `req.url` is exactly `/phrase`.

```
req.url ~ "phrase"
```

Matches `phrase` anywhere.

Matching at the beginning or end of a string

```
req.http.host ~ "^www"
```

Matches if `req.http.host` starts with `www`.

```
req.url ~ "\.jpg$"
```

Matches if `req.url` ends with `.jpg`.

Multiple matches

```
req.url ~ "\.(png|jpg|css|js)$"
```

Matches if `req.url` ends with either `.png`, `.jpg`, `.css`, or `.js`.

```
req.url ~ "\.php(\?.*)?$"
```

Matches if `req.url` ends with `.php`, `.php?foo=bar` or `.php?`, but not `.phpa`.

ℹ **NOTE:** You can also use `req.url.ext` to find the file extension specified in a URL. For example, in the request `www.example.com/1/hello.gif?foo=bar`, `req.url.ext` will return `gif`. See our Miscellaneous VCL features (/guides/vcl/miscellaneous-VCL-extensions) guide for more information.

```
req.url ~ "\.[abc]server$"
```

Matches if `req.url` ends with `.aserver`, `.bserver` or `.cserver`.

Matching wildcards

```
req.url ~ "jp.g$"
```

Matches if `req.url` ends with `jpeg`, `jpag`, and `jp0g`, but doesn't match if `req.url` ends with `jpg`. It also matches if any other character is between the `jp` and the `g`.

```
req.url ~ "jp.*g$"
```

Matches `jp` followed by 0 or more random characters ending with the letter `g` (`jpeg`, `jpg`, and `jpeeeeg` all match).

Capturing matches

```
set req.http.Foo = "abbbccccc";
if (req.http.Foo ~ "^(a+)(b+)(c+)") {
  set resp.http.match0 = re.group.0; # now equals 'abbbccccc'
  set resp.http.match1 = re.group.1; # now equals 'a'
  set resp.http.match2 = re.group.2; # now equals 'bbb'
  set resp.http.match3 = re.group.3; # now equals 'cccccc'
}
```

The `re.group.[0-9]` objects allow you to capture matches. The `re.group.0` object evaluates to the entire matched string even if no capture groups have been supplied. You can use these objects to replace this example:

```
if (req.url ~ "(?i)\?.*some_query_arg=([^&]*)") {
  set req.http.Thing-I-Want = regsub(req.url, "(?i)\?.*some_query_arg=([^&]*).*", "\1");
}
```

You can use `re.group` to greatly simplify the previous example:

```
if (req.url ~ "(?i)\?.*some_query_arg=([^&]*)") {
  set req.http.Thing-I-Want = re.group.1;
}
```

You could even get really fancy and do something like this:

```
set req.http.Thing-I-Want = if(req.url ~ "(?i)\?.*some_query_arg=([^&]*)", re.group.1, "");
```

Replacing content

```
set req.http.host = regsub(req.http.host, "^www\.","");
```

Removes a leading `www.` from the host, if present.

```
set req.http.api-test = regsub(req.http.host, "^www.","api.");
```

Sets the `api-test` header to contain the host-header, but replaces a leading `www.` with `api`:

```
Host: www.example.com ->
Host: www.example.com
api-test: api.example.com
Host: example.com ->
Host: example.com
api-test: example.com
```

```
set req.url = regsuball(req.url, "/+", "/");
```

Changes all occurrences of multiple slashes in the URL to a single slash. For example, `//docs///intro.html` will be transformed to `/docs/intro.html`.

---

§ About VCL Snippets (/guides/vcl-snippets/about-vcl-snippets)

IMPORTANT: This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

VCL Snippets are short blocks of VCL logic (/guides/vcl/guide-to-vcl) that can be included directly in your service configurations. They're ideal for adding small sections of code when you don't need more complex, specialized configurations that sometimes require custom VCL (/guides/vcl/uploading-custom-vcl). Fastly supports two types of VCL Snippets:

- **Regular VCL Snippets (/guides/vcl-snippets/using-regular-vcl-snippets)** get created as you create versions of your Fastly configurations. They belong to a specific service and any modifications you make to the snippet are locked and deployed when you deploy a new

version of that service. You can treat regular snippets like any other Fastly objects because we continue to clone them and deploy them with a service until you specifically delete them. You can create regular snippets using either the web interface or via the API.

- **Dynamic VCL Snippets (/guides/vcl-snippets/using-dynamic-vcl-snippets)** can be modified and deployed any time they're changed. Because they are versionless objects (much like Edge Dictionaries (/guides/edge-dictionaries/) or Edge ACLs (/guides/access-control-lists/)), dynamic snippets can be modified independently from service changes. This means you can modify snippet code rapidly without deploying a service version that may not be ready for production. You can only create dynamic snippets via the API.

Limitations of VCL Snippets

- Snippets are limited to 1MB in size by default. If you need to store snippets larger than the limit, contact support@fastly.com (mailto:support@fastly.com).
- Snippets do not currently support conditions, though `if` statements can be used within snippet code instead.
- Snippets cannot currently be shared between services.

---

§ Using dynamic VCL Snippets (/guides/vcl-snippets/using-dynamic-vcl-snippets)

Dynamic VCL Snippets are one of two types of snippets (/guides/vcl-snippets/about-vcl-snippets) that allow you to insert small sections of VCL logic into your service configuration without requiring custom VCL (/guides/vcl/uploading-custom-vcl) (though you can still include snippets in custom VCL when necessary).

You can only create dynamic snippets via the API. Because they are versionless objects (much like Edge Dictionaries (/guides/edge-dictionaries/) or Edge ACLs (/guides/access-control-lists/)), dynamic snippets can be modified independently from changes to your Fastly service. This means you can modify snippet code rapidly without deploying a service version that may not be ready for production.

Creating and using a dynamic VCL Snippet

Using the cURL command line tool, make the following API call in a terminal application:

```
curl -X POST -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snippe
t -H "Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: application/x-www-form-urlencoded' --data
$'name=my_dynamic_snippet_name&type=recv&dynamic=1&content=if ( req.url ) {\n set req.http.my-s
nippet-test-header = "true";\n}');
```

Fastly returns a JSON response that looks like this:

```
{
 "service_id": "<Service Id>",
 "version": "<Editable Version>",
 "name": "my_dynamic_snippet_name",
 "type": "recv",
 "priority": 100,
 "dynamic": 1,
 "content": null,
 "id": "decafbad12345",
 "created_at": "2016-09-09T20:34:51+00:00",
 "updated_at": "2016-09-09T20:34:51+00:00",
 "deleted_at": null
}
```

ℹ **NOTE:** The returned JSON includes `"content": null`. This happens because the content is stored in a separate, unversioned object.

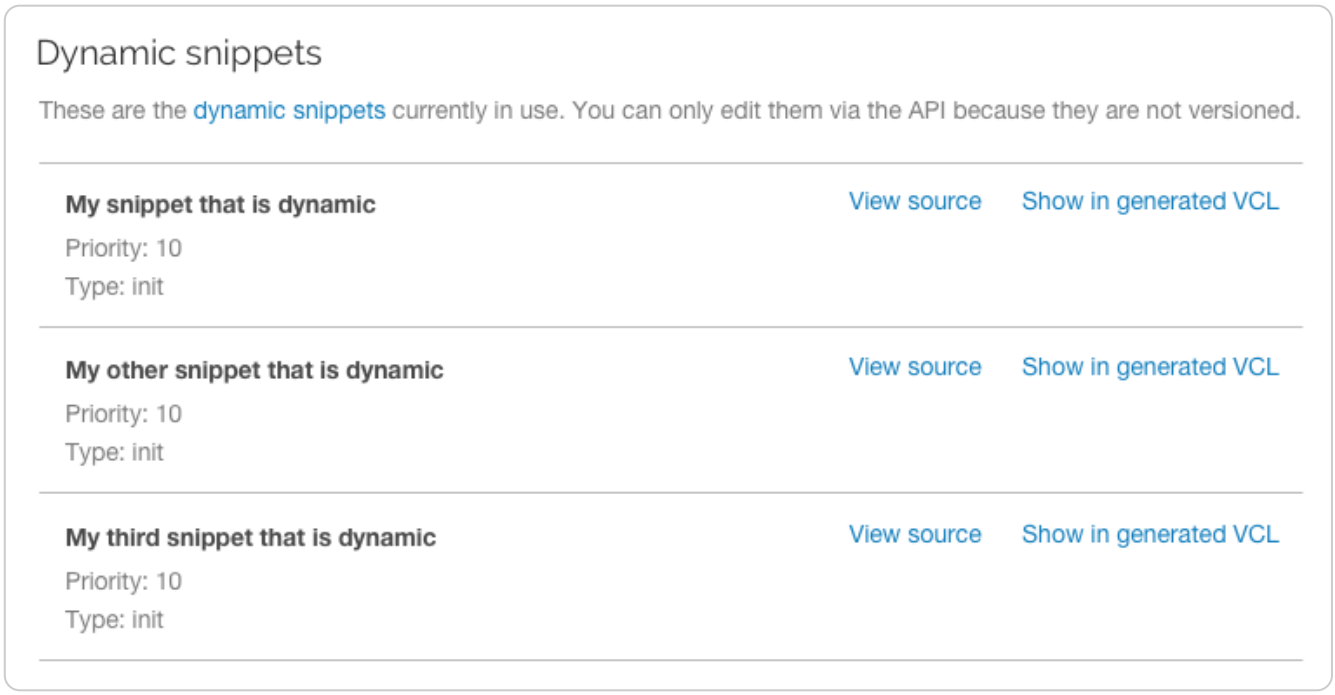
Viewing dynamic VCL Snippets in the web interface

You can view a list of dynamic VCL snippets. You can also view just the source of a specific snippet or a specific snippet's location in generated VCL.

Viewing a list of dynamic VCL Snippets

To view the entire list of a service's dynamic VCL Snippets directly in the web interface:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears listing all dynamic VCL Snippets for your service in the Dynamic snippets area.



Viewing the source of a specific snippet

You can view just the source of a specific snippet:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears.
4. Click the **View Source** link to the right of the name of the snippet. A view source window appears.

Viewing the location of a specific snippet in generated VCL

You can view a specific snippet's location in generated VCL:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears.
4. Click the **Show in Generated VCL** link to the right of the name of the snippet. The Generated VCL window appears.

Fetching a list of all dynamic VCL Snippets

To list all dynamic VCL Snippets attached to a service, make the following API call in a terminal application:

```
curl -X GET -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snippet
-H "Fastly-Key:FASTLY_API_TOKEN"
```

Fetching an individual dynamic VCL Snippet

To fetch an individual snippet, make the following API call in a terminal application:

```
curl -X GET -s https://api.fastly.com/service/<Service ID>/snippet/<my_dynamic_snippet_name> -H
"Fastly-Key:FASTLY_API_TOKEN"
```

Unlike fetching regular VCL Snippets (/guides/vcl-snippets/using-regular-vcl-snippets#fetching-an-individual-regular-vcl-snippet), you do not include the version in the URL and you must use the ID returned when the snippet was created, not the name.

Updating an existing dynamic VCL Snippet

To update an individual snippet, make the following API call in a terminal application:

```
curl -X PUT -s https://api.fastly.com/service/<Service ID>/snippet/<my_dynamic_snippet_name> -H
"Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: application/x-www-form-urlencoded' --data $'con
tent=if ( req.url ) {\n set req.http.my-snippet-test-header = \"affirmative\";\n}';
```

Deleting an existing dynamic VCL Snippet

To delete an individual snippet, make the following API call in a terminal application:

```
curl -X DELETE -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snip
pet/<my_dynamic_snippet_name> -H "Fastly-Key:FASTLY_API_TOKEN"
```

Including dynamic snippets in custom VCL

By specifying a location of `none` for the `type` parameter, snippets will not be rendered in VCL. This allows you to include snippets in custom VCL using the following syntax:

`include "snippet::<snippet name>"`

The same VCL Snippet can be included in custom VCL in as many places as needed.

Example use: blocking site scrapers

Say you wanted to implement some pattern matching against incoming requests to block someone trying to scrape your site. Say also that you've developed a system that looks at all incoming requests and generates a set of rules that can identify scrapers using a combination of the incoming IP address, the browser, and the URL they're trying to fetch. Finally, say that the system updates the rules every 20 minutes.

If, during system updates, your colleagues are also making changes to the rest of your Fastly configuration, you probably don't want the system to automatically deploy the latest version of the service since it might be untested. Instead you could generate the rules as a Dynamic VCL Snippet. Whenever the snippet is updated, all other logic remains the same as the currently deployed version and only your rules are modified.

§ Using regular VCL Snippets (/guides/vcl-snippets/using-regular-vcl-snippets)

Regular VCL Snippets are one of two types of snippets (/guides/vcl-snippets/about-vcl-snippets) that allow you to insert small sections of VCL logic into your service configuration without requiring custom VCL (/guides/vcl/uploading-custom-vcl) (though you can still include snippets in custom VCL when necessary).

Unlike dynamic snippets (/guides/vcl-snippets/using-dynamic-vcl-snippets), regular snippets can be created via the web interface or via the API. They are considered "versioned" objects. They belong to a specific service and any modifications you make to the snippet are locked and deployed when you deploy a new version of that service. We continue to clone them and deploy them with a service until you specifically delete them.

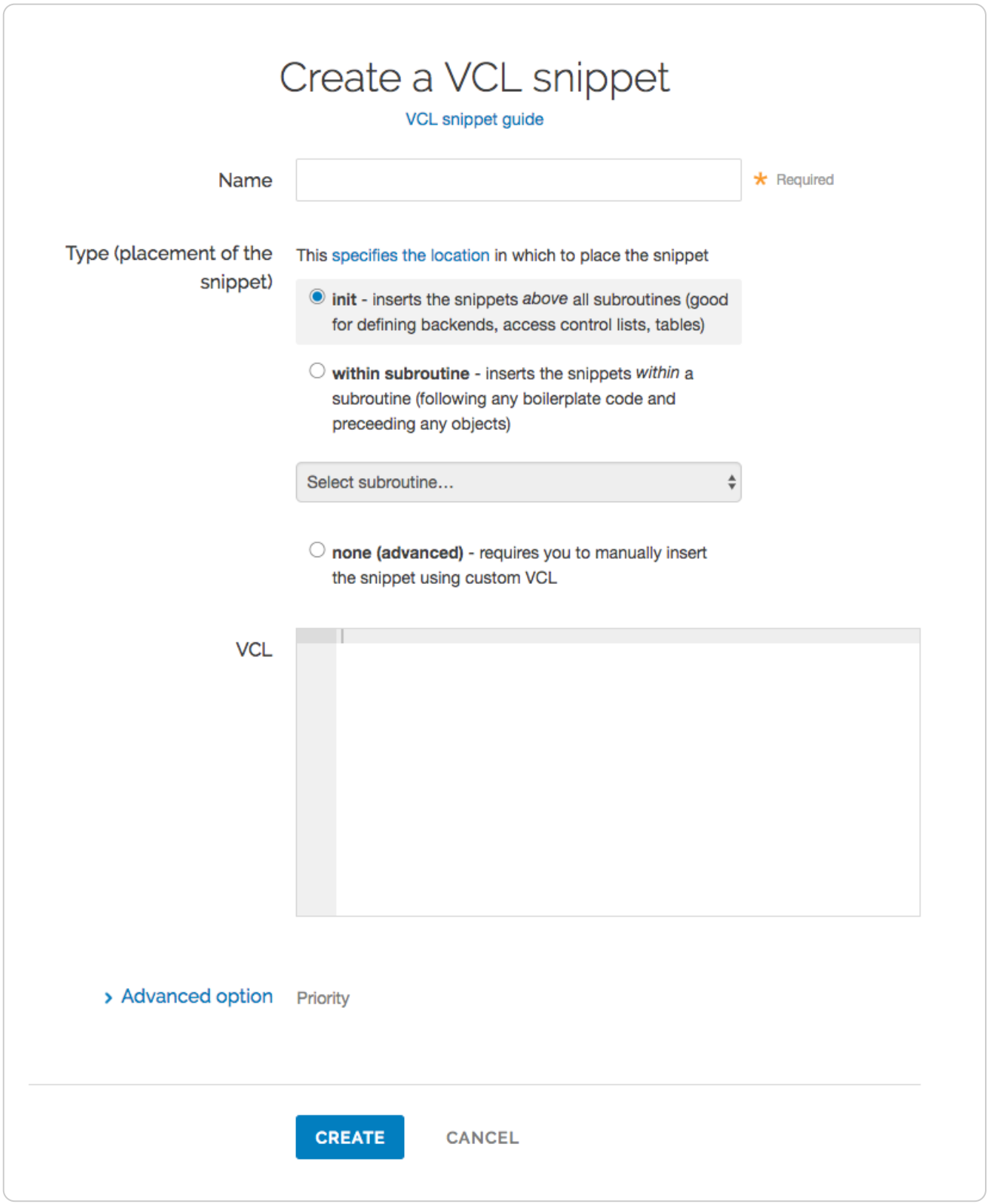
Creating a regular VCL Snippet

You can create regular VCL Snippets via the web interface or via the API.

Via the web interface

To create a regular VCL Snippet via the web interface:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears.
4. Click **Create Snippet**. The Create a VCL snippet page appears.



5. In the **Name** field, type an appropriate name (for example, `Example Snippet`).
6. Using the **Type** controls, select the location in which the snippet should be placed as follows:
   - Select `init` to insert it above all subroutines in your VCL.

- Select `within subroutine` to insert it within a specific subroutine and then select the specific subroutine from the **Select subroutine** menu.
- Select `none (advanced)` to insert it manually. See Including regular snippets in custom VCL (/guides/vcl-snippets/using-regular-vcl-snippets#including-regular-snippets-in-custom-vcl) for the additional manual insertion requirements if you select this option.

7. In the **VCL** field, type the snippet of VCL logic to be inserted for your service version.
8. Click **Create** to create the snippet.

Via the API

To create a regular VCL Snippet via the API, make the following API call using the cURL command line tool in a terminal application:

```
curl -X POST -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snippe
t -H "Fastly-Key:FASTLY_API_TOKEN" -H `fastly-cookie` -H 'Content-Type: application/x-www-form-
urlencoded' --data $'name=my_regular_snippet&type=recv&dynamic=0&content=if ( req.url ) {\n set
req.http.my-snippet-test-header = "true";\n}';
```

Fastly returns a JSON response that looks like this:

```
{
 "service_id": "<Service Id>",
 "version": "<Editable Version>",
 "name": "my_regular_snippet",
 "type": "recv",
 "content": "if ( req.url ) {\n set req.http.my-snippet-test-header = \"true\";\n}",
 "priority": 100,
 "dynamic": 0,
 "id": "56789exampleid",
 "created_at": "2016-09-09T20:34:51+00:00",
 "updated_at": "2016-09-09T20:34:51+00:00",
 "deleted_at": null
}
```

NOTE: When regular VCL snippets get created, an `id` field will be returned that isn't used. The field only applies to dynamic VCL Snippets (/guides/vcl-snippets/using-dynamic-vcl-snippets). In addition, the returned JSON includes a populated `content` field because the snippet content is stored in a versioned object.

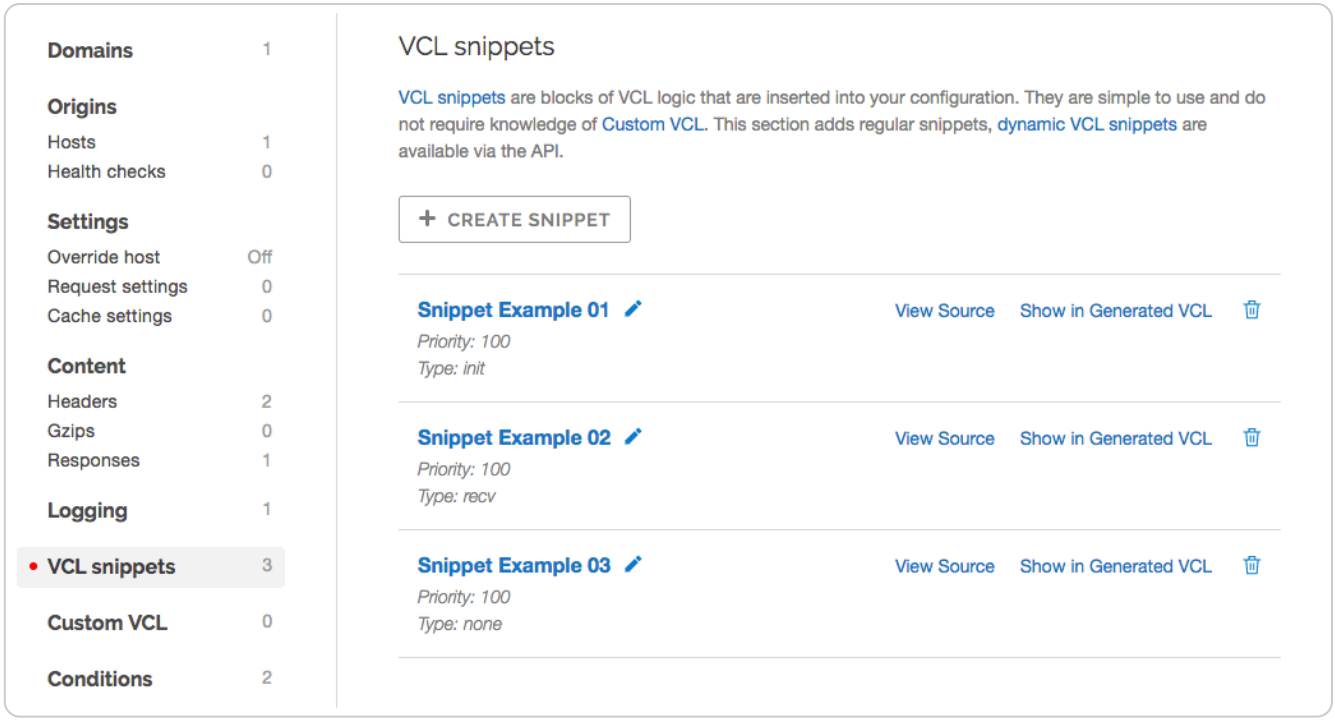
Viewing regular VCL Snippets in the web interface

You can view a list of regular VCL snippets. You can also view just the source of a specific snippet or a specific snippet's location in generated VCL.

Viewing a list of regular VCL Snippets

To view the entire list of a service's regular VCL Snippets directly in the web interface:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears listing all available VCL snippets for your service.



Viewing the source of a specific snippet

You can view just the source of a specific snippet:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears.
4. Click the **View Source** link to the right of the name of the snippet. A view source window appears.

Viewing the location of a specific snippet in generated VCL

You can view a specific snippet's location in generated VCL:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears.

4. Click the **Show in Generated VCL** link to the right of the name of the snippet. The Generated VCL window appears.

Fetching regular VCL Snippets via the API

You can fetch regular VCL Snippets for a particular service via the API either singly or all at once.

Fetching an individual regular VCL Snippet

To fetch an individual snippet, make the following API call in a terminal application:

```
curl -X GET -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snippe
t/<Snippet Name e.g my_regular_snippet> -H "Fastly-Key:FASTLY_API_TOKEN"
```

Unlike fetching dynamic VCL Snippets (/guides/vcl-snippets/using-dynamic-vcl-snippets#fetching-an-individual-dynamic-vcl-snippet) you include the version in the URL and you must use the name of the snippet, not the ID.

Fetching a list of regular VCL Snippets

To list all regular VCL Snippets attached to a service, make the following API call in a terminal application:

```
curl -X GET -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snippe
t/ -H "Fastly-Key:FASTLY_API_TOKEN"
```

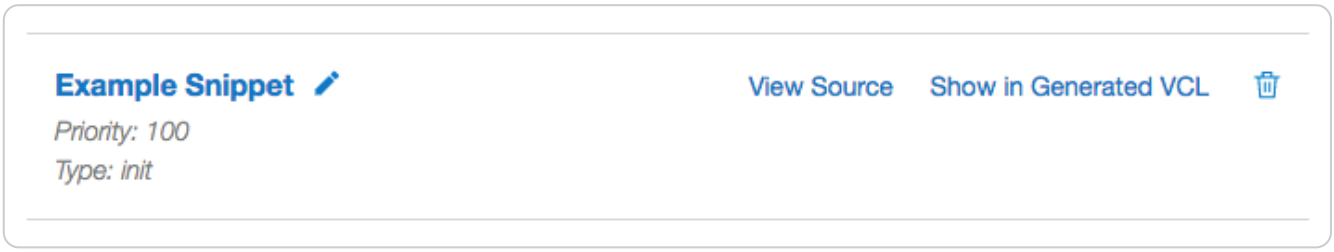
Updating an existing regular VCL Snippet

You can update existing regular VCL Snippets via the web interface or via the API.

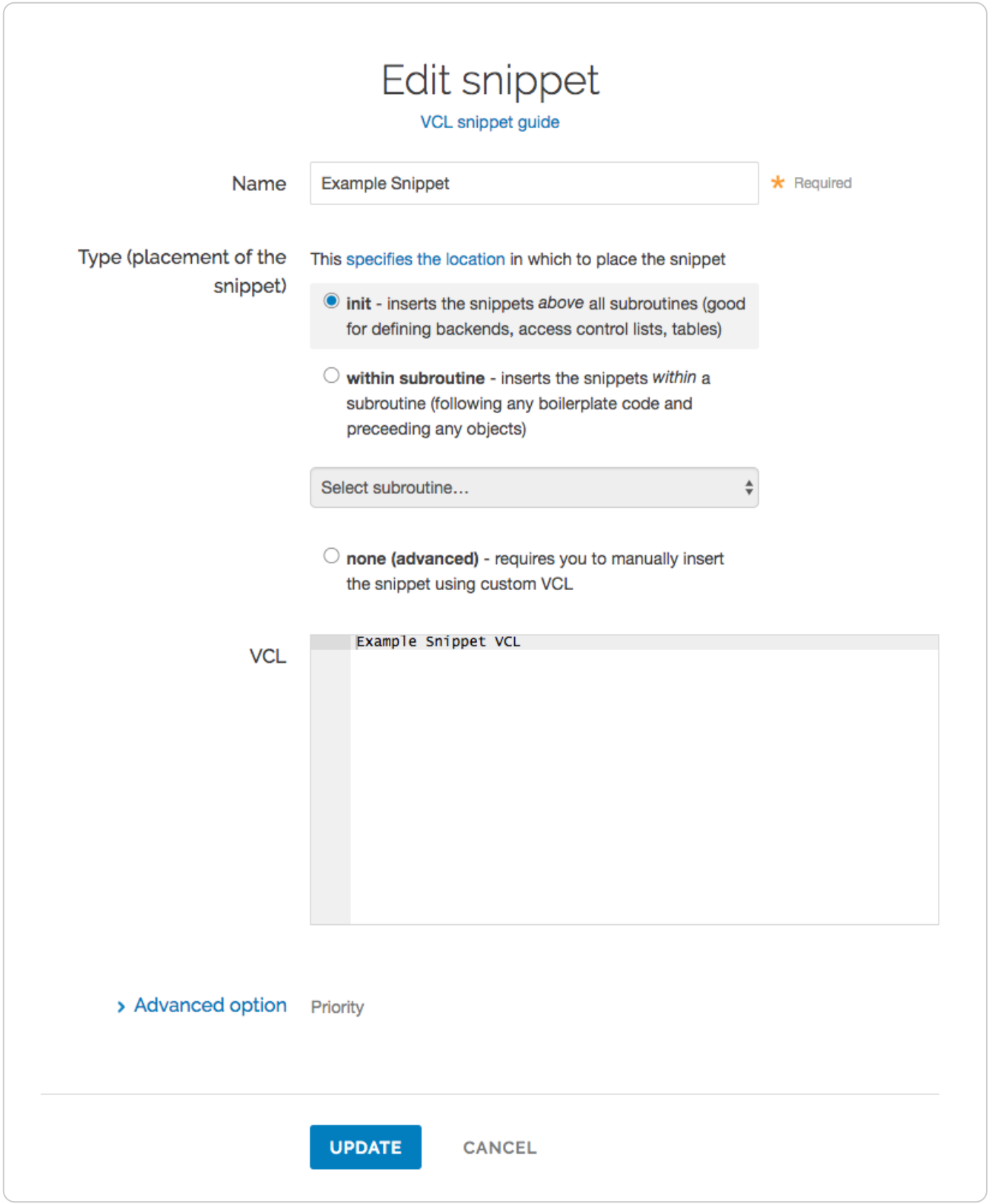
Via the web interface

To update an individual snippet via the web interface:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears.
4. Click the pencil icon next to the name of the snippet to be updated.



The Edit snippet page appears.



5. Update the snippet's settings or VCL as appropriate.
6. Click **Update** to save your changes.

Via the API

To update an individual snippet via the API, make the following API call in a terminal application:

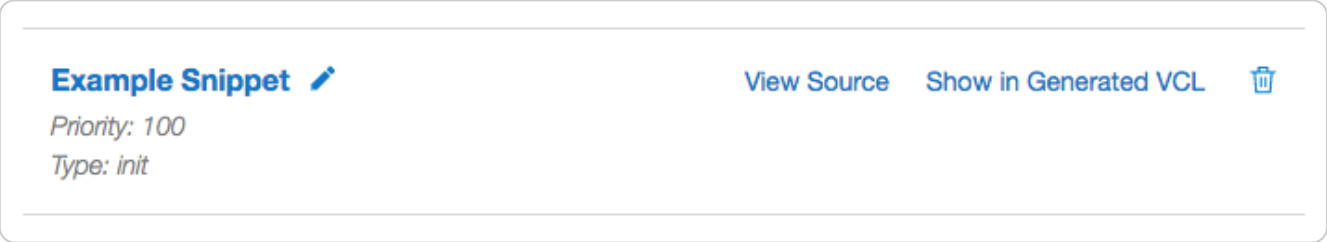
```
curl -X PUT -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snippe
t/<Snippet Name e.g my_regular_snippet> -H "Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: appl
ication/x-www-form-urlencoded' --data $'content=if ( req.url ) {\n set req.http.my-snippet-test
-header = \"affirmative\";\n}';
```

Deleting an existing regular VCL Snippet

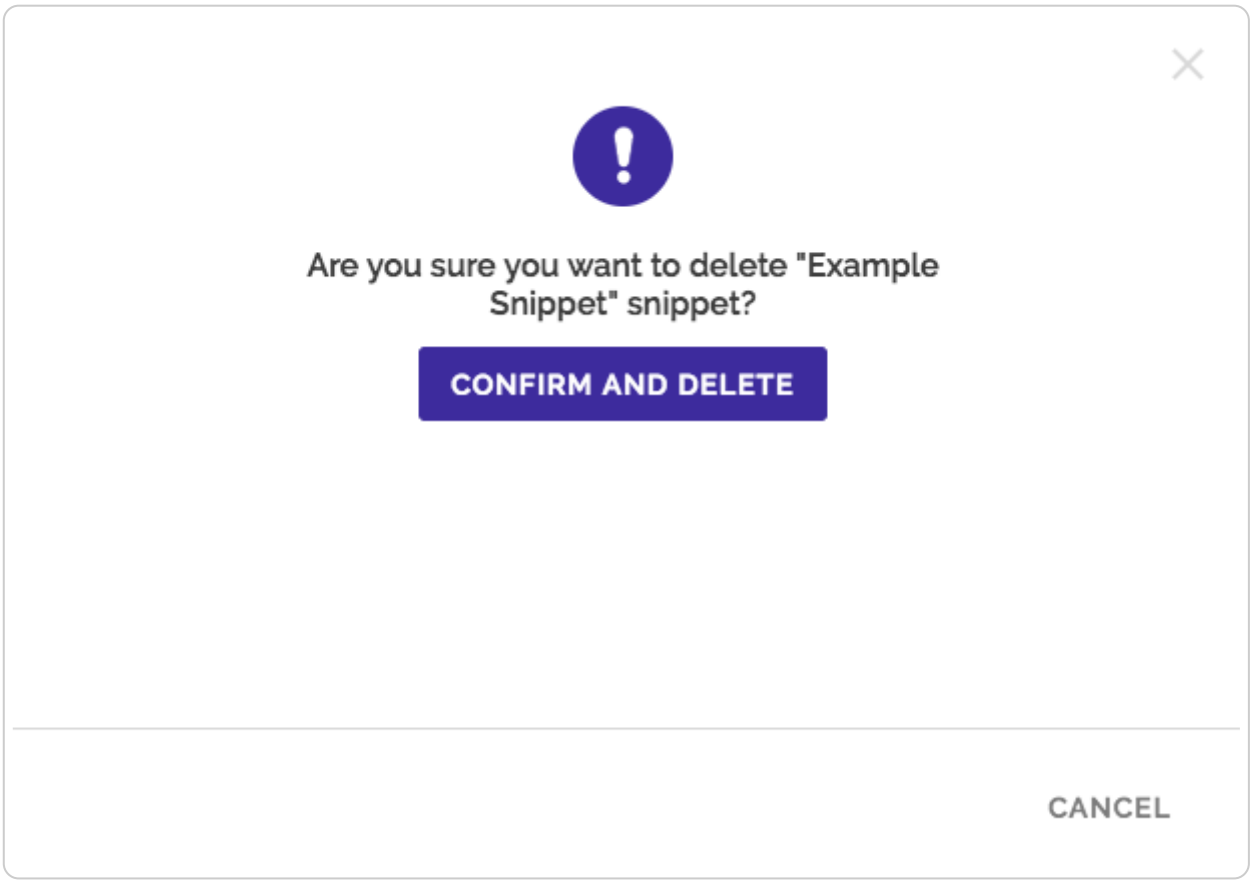
You can update existing regular VCL Snippets via the web interface or via the API.

Via the web interface

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **VCL Snippets** link. The VCL Snippets page appears.
4. Click the trashcan icon to the right of the name of the snippet to be updated.



A confirmation window appears.



5. Click **Confirm and Delete**.

Via the API

To delete an individual snippet via the API, make the following API call in a terminal application:

```
curl -X DELETE -s https://api.fastly.com/service/<Service ID>/version/<Editable Version #>/snippet/<Snippet Name e.g my_regular_snippet> -H "Fastly-Key:FASTLY_API_TOKEN"
```

Including regular snippets in custom VCL

Snippets will not be rendered in VCL if you select `none (advanced)` for the snippet type in the web interface or specify a location of `none` for the `type` parameter in the API. This allows you to manually include snippets in custom VCL using the following syntax:

`include "snippet::<snippet name>"`

The same VCL Snippet can be included in custom VCL in as many places as needed.

Example use: location-based redirection

Say that you work at a large content publisher and you want to redirect users to different editions of your publication depending on which country their request comes from. Say also that you want the ability to override the edition you deliver to them based on a cookie.

Using regular VCL snippets, you could add a new object with the relevant VCL as follows:

```
if (req.http.Cookie:edition == "US" || client.geo.country_code == "US" || ) {
  set req.http.Edition = "US";
  set req.backend = F_US;
} elseif (req.http.Cookie:edition == "Europe" || server.region ~ "^EU-" ) {
  set req.http.Edition = "EU";
  set req.backend = F_European;
} else {
  set req.http.Edition = "INT";
  set req.backend = F_International;
}
```

This would create an Edition header in VCL, but allow you to override it by setting a condition. You would add the Edition header into Vary (https://www.fastly.com/blog/best-practices-for-using-the-vary-header) and then add a false condition (/guides/conditions/using-conditions#using-operators-to-perform-matches-on-complex-logical-expressions) (e.g., `!reg.url`) to your other backends to ensure the correct edition of your publication gets delivered (Remember: VCL Snippets get added to VCL before backends are set.)

- Guides (/guides/) > Developer's tools > Tutorials (/guides/tutorials/)

§ Basic authentication (/guides/tutorials/basic-authentication)

Basic authentication (https://en.wikipedia.org/wiki/Basic_access_authentication) is a simple way of protecting a website at the edge. Users enter a username and password combination to access pages protected by basic authentication. You can use basic authentication to restrict access to low-risk assets like testing and staging environments. Basic authentication can be implemented using custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl#fastlys-vcl-boilerplate) or VCL Snippets (/guides/vcl-snippets/using-regular-vcl-snippets#creating-a-regular-vcl-snippet).

⚠ **WARNING:** Basic authentication shouldn't be used to restrict access to sensitive information. See the security considerations section for more information.

Follow the steps below to set up basic authentication for your service:

1. Create an Edge Dictionary (/guides/edge-dictionaries/creating-and-using-dictionaries) with a list of Base64-encoded usernames and passwords — you can include the usernames in plaintext for reference. You can use the API (/guides/edge-dictionaries/creating-and-using-dictionaries#create-an-empty-dictionary-container-within-a-service) to create the Edge Dictionary and add dictionary items, or you can use custom VCL (/guides/edge-dictionaries/creating-and-using-dictionaries#custom-vcl-examples) as shown below.

   ```
   table customer_keys {
     "Basic am9lOjQzNEAvMzkyIzgyPzk2": "joe",
     "Basic bWlrZTo4MjM0MzNzWjQ0SDZlNw==": "mike"
   }
   ```

   The first value in the key pair is the username and password Base64-encoded. You can generate this in a terminal application as shown below. In this example, the username is `joe`, and the password is `434@/392#82?96`.

   ```
   $ echo -ne joe:434@/392#82?96 | base64
   am9lOjQzNEAvMzkyIzgyPzk2
   ```

   The result (`am9lOjQzNEAvMzkyIzgyPzk2`) is the second half of the first key pair (`Basic am9lOjQzNEAvMzkyIzgyPzk2`).

2. In `vcl_recv`, create a table lookup to authorize customer credentials against those in the table.

   ```
   ##table lookup from customer_keys dictionary, plus part in vcl_error
   if(! table.lookup(customer_keys, req.http.Authorization) ) {
     error 401 "Restricted";
   }
   ```

3. In `vcl_error`, create your Custom 401 Restricted HTML page.

```
## Start 401 custom code
if (obj.status == 401) {
  set obj.http.Content-Type = "text/html; charset=utf-8";
  set obj.http.WWW-Authenticate = "Basic realm=Secured";
  synthetic {"

 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">

 <HTML>
 <HEAD>
 <TITLE>Error</TITLE>
 <META HTTP-EQUIV='Content-Type' CONTENT='text/html;'>
 </HEAD>
 <BODY><H1>401 Unauthorized (varnish)</H1></BODY>
 </HTML>
 "};
  return (deliver);
} # End custom 401 code
```

Using basic authentication with GCS

To use basic authentication with Google Cloud Storage (/guides/integrations/google-cloud-storage) (GCS) as a origin server, add a request header (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) to delete the `http.Authorization` header and prevent it from being sent to GCS. That header causes GCS to respond with a "Not Authorized" message instead of your request.

Security considerations

There are several security considerations you should take into account before using basic authentication:

- Basic authentication can't protect high-risk information. Don't use it to restrict access to sensitive information.
- If you're not using TLS (/guides/detailed-product-descriptions/tls-service-options), the password will be transmitted over the wire in Base64 encoding. The encoded string could easily be captured using an application like Wireshark and converted to plaintext.
- The password is cached by the user's web browser, and it can be permanently saved by the user's web browser.

Using access control lists

As an alternative to basic authentication, you can use access control lists (ACLs) to restrict access to your assets by whitelisting a set of IP addresses. To whitelist IP addresses with an ACL, add custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl#fastlys-vcl-boilerplate) to Fastly's boilerplate VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl#inserting-custom-vcl-in-fastlys-vcl-boilerplate).

```
# Who is allowed access ...
acl local {
    "localhost";
    "192.168.1.0"/24; /* and everyone on the local network */
    ! "192.168.1.23"; /* except for the dial-in router */
}
```

See our ACL guides (/guides/access-control-lists/about-acls) for more information.

---

§ Cache control tutorial (/guides/tutorials/cache-control-tutorial)

You are in full control of how Fastly caches your resources. The most preferred way of instructing Fastly is to use backend HTTP headers. The other way is to use the Varnish Configuration Language (/guides/vcl/guide-to-vcl) (VCL).

❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Backend HTTP headers

You can set four different types of HTTP headers which will have different effects on our caches and on web browsers. If you use more than one type, they are prioritized in the order listed below:

Surrogate-Control

Format:

```
Surrogate-Control: max-age=(time in seconds)
```

Example:

```
Surrogate-Control: max-age=3600
```

will cache something on our caches for one hour. This header gets stripped and is only visible to Fastly caches.

Cache-Control: s-maxage

Format:

```
Cache-Control: s-maxage=(time in seconds)
```

This is the same as `Surrogate-Control`, except the header is not stripped and will be respected by Fastly caches and any caches between Fastly and the browser, but not the browser itself.

Cache-Control: max-age

Format:

```
Cache-Control: max-age=(time in seconds)
```

This header will be respected by Fastly caches, any caches between Fastly and the browser, and the browser itself.

Expires

This header caches content until it expires as specified. It will be respected by Fastly caches, any caches between Fastly and the browser and the browser itself. See section 5.3 of RFC7234 (https://tools.ietf.org/html/rfc7234#section-5.3) for an explanation of the format.

Do not cache

If you want to ensure that a resource is not cached by Fastly, send the following HTTP header with the origin response:

```
Cache-Control: private
```

If you just set `max-age=0` or an `Expires` in the past, Fastly may still use a single response to satisfy multiple outstanding requests that arrive while waiting on the origin (see Request collapsing (/guides/performance-tuning/request-collapsing)), or may cache the object in a stale form so that it can be used in case of errors or asyncronous revaldiation (see Serving stale content (/guides/performance-tuning/serving-stale-content)).

The `private` directive does not prevent content from being cached in the browser. If you need to prevent caching by both Fastly *and* web browsers, we recommend combining the `private` directive with `max-age=0` or `no-store`. For example:

```
Cache-Control: private, no-store
```

Fastly does not currently respect `no-store` or `no-cache` directives. Including either or both of these in a `Cache-Control` header has no effect on Fastly's caching decision, unless you alter this behavior using custom VCL.

When Cache-Control and Surrogate-Control headers co-exist

Say that you want Fastly to cache your resources forever but send headers to browsers so that they don't cache it at all (so that every browser miss hits Fastly but isn't a cache miss from your service).

The best way to do this would be to send Fastly both the `Cache-Control` header as you want it to go to the browsers, and use `Surrogate-Control` to tell us how long to cache for. For example:

```
Cache-Control: no-cache, no-store, must-revalidate
Surrogate-Control: max-age=3600
Pragma: no-cache
Expires: 0
```

Except for when the `Cache-Control` header is set to `private`, the `Surrogate-Control` header takes priority over `Cache-Control`, but unlike `Cache-Control` it is stripped so the browsers don't see it.

Example backend configs

Apache Config

If you are using Apache, the easiest way to add headers is to use the mod_expires module (http://httpd.apache.org/docs/current/mod/mod_expires.html). For example, to cache GIF images for 75 minutes after the image was last accessed by the cache server, you would add a directive like this under the VirtualHost (or globally). For example:

```
ExpiresActive On
ExpiresByType image/gif "access plus 1 hours 15 minutes"
```

You can also cache whole URL subtrees. For example:

```
<Location "/css">
  ExpiresActive On
  ExpiresDefault "access plus 1 year"
</Location>

<Location "/static/">
  ExpiresActive On
  ExpiresDefault "access plus 1 day"
</Location>
```

NGINX Configuration

To configure NGINX, add the expires directive. For example:

```
location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
   expires 1h;
}
```

Alternatively, if you need more flexibility in modifying headers you can try the HttpHeadersMore Module (http://nginx.org/en/docs/http/ngx_http_headers_module.html).

Amazon S3 configuration

By default, S3 doesn't have a facility for setting Cache-Control headers across multiple objects, so you will have to do this file-by-file using the S3Explorer, or in an automated fashion by using a cloud library like boto. Remember that you can combine long cache time with instant purges to enhance your performance.

★ **TIP:** While it's difficult to get S3 to set `Surrogate-Control`, you can set `x-amz-meta-surrogate-control` (/guides/purging/setting-surrogate-key-headers-for-amazon-s3-origins) instead on origin and Fastly will honor that.

```
from boto.s3.connection import S3Connection

connection = S3Connection('aws access key', 'aws secret key')

buckets = connection.get_all_buckets()

for bucket in buckets:
    for key in bucket.list():
        print('%s' % key)

        if key.name.endswith('.jpg'):
            contentType = 'image/jpeg'
        elif key.name.endswith('.png'):
            contentType = 'image/png'
        else:
            continue

        key.metadata.update({
            'Content-Type': contentType,
            'Cache-Control': 'max-age=864000'
        })
        key.copy(
            key.bucket.name,
            key.name,
            key.metadata,
            preserve_acl=True
        )
```

❗ **IMPORTANT:** The above example provides an S3 configuration option for customers with small- to medium-sized buckets. However, it iterates over every object in those buckets. If you have millions of objects this may not be the right approach. For millions of objects, we recommend using VCL (/guides/vcl/guide-to-vcl). Be sure to contact us (mailto:support@fastly.com) for assistance.

Custom Headers in Programming Languages and Frameworks

PHP

More information: http://php.net/manual/en/function.header.php (http://php.net/manual/en/function.header.php)

Example: add this to your PHP code before you send any output to cache certain HTML for an hour

```
header('Cache-Control: max-age=3600');
```

Django

More information: https://docs.djangoproject.com/en/dev/ref/request-response/#setting-headers (https://docs.djangoproject.com/en/dev/ref/request-response/#setting-headers)

Example:

```
response = HttpResponse()
response['Cache-Control'] = 'max-age=3600'
```

Sinatra

More information: http://sinatra.rubyforge.org/doc/ (http://sinatra.rubyforge.org/doc/)

Example:

```
get '/' do
    headers['Cache-Control'] = 'max-age=3600'
end
```

★ **TIP:** Expiration times in these examples are provided for guidance only. You can use longer expirations coupled with our purging API (/api/purge) to make your site faster and your backend less loaded.

---

§ Enabling URL token validation (/guides/tutorials/enabling-url-token-validation)

Token validation allows you to create URLs that expire. Tokens are generated within your web application and appended to URLs in a query string. Requests are authenticated at Fastly's edge instead of your origin server. When Fastly receives a request for the URL, the token is validated before serving the content. After a configurable period of time, the token expires.

Adding custom VCL

To enable token validation, you'll need to create a Varnish configuration (/guides/vcl/uploading-custom-vcl) named `vcl_recv` and add the following example code to it.

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

```
/* make sure there is a token */
if (req.url !~ ".+\?.*token=(\d{10,11})_([^&]+)") {
  error 403;
}

/* extract token expiration and signature */
set req.http.X-Exp = re.group.1;
set req.http.X-Sig = re.group.2;

/* validate signature */
if (req.http.X-Sig == regsub(digest.hmac_sha1(digest.base64_decode("YOUR%SECRET%KEY%IN%BASE64%H
ERE"),
req.url.path req.http.X-Exp), "^0x", "")) {

  /* check that expiration time has not elapsed */
  if (time.is_after(now, std.integer2time(std.atoi(req.http.X-Exp)))) {
    error 410;
  }

} else {
  error 403;
}

/* cleanup variables */
unset req.http.X-Sig;
unset req.http.X-Exp;
```

WARNING: You must replace `YOUR%SECRET%KEY%IN%BASE64%HERE` in the example VCL with your own randomly generated secret key. The example key will intentionally cause an error if you use it. Anyone who learns this key can bypass your token validation, so it's critical that you keep this key secret.

The key found in `digest.hmac_sha1` can be any string. The one in this example was generated with the command `openssl rand -base64 32`.

A token is expected in the `?token=` GET parameter. Tokens take the format `[expiration]_[signature]` and look like this:

```
1441307151_4492f25946a2e8e1414a8bb53dab8a6ba1cf4615
```

The full request URL looks like this:

```
http://www.example.com/foo/bar.html?token=1441307151_4492f25946a2e8e1414a8bb53dab8a6ba1cf4615
```

What the custom VCL does

The custom VCL code checks for two things:

- Is the current time greater than the expiration time specified in the token?
- Does our signature match the signature of the token?

If the signature is invalid, Varnish returns a 403 response. If the signature is valid but the expiration time has elapsed, Varnish returns a 410 response. The different response codes are helpful for debugging.

Configuring your application

You'll need to write custom code in your application to generate tokens and authenticate with Varnish. We provide examples in our token functions (https://github.com/fastly/token-functions) repository on GitHub. Review the examples in the repository to learn how to generate custom tokens within your application.

Testing

To test your configuration, append a token generated by your application to a URL in a query string. For example:

`http://www.example.com/foo/bar.html?token=1441307151_4492f25946a2e8e1414a8bb53dab8a6ba1cf4615`

If the token is valid, you will receive a normal response. If it is invalid, you will receive a 403 response.

Troubleshooting NUL bytes

You should verify that your secret key is devoid of NUL bytes. If the base64-decoded string contains a NUL byte (0x00), then that byte and any bytes following it will not be included in the response. See base64 decoding (/guides/vcl/cryptographic-and-hashing-related-vcl-functions#base64-decoding) for more information.

• Guides (/guides/) > Diagnostics and performance > Streaming logs (/guides/streaming-logs/)

§ Changing log line formats (/guides/streaming-logs/changing-log-line-formats)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature allows you to change the format that your log messages are delivered in. By default, we send log messages out in standard syslog format. The prefix for this format (as defined in RFC 3164 (https://tools.ietf.org/html/rfc3164)) appears as follows:

```
<134>2016-07-04T22:37:26Z cache-sjc3128 LogTest[62959]: <your log message>
```

The prefix begins with the message priority (always `<134>`, which means `Facility=Local0, Severity=Informational`), followed the date and time the log was sent (`2016-07-04T22:37:26Z`), the cache node it came from (in this case `cache-sjc3128`), the name of your log (`LogTest`) and the ID of the process sending it (`62959`).

Available message formats

Although the default message prefix works for most logging services and processors, we allow you to choose one of several formats:

- `classic` is the default prefix format. A standard syslog prefix as defined by RFC 3164 (https://tools.ietf.org/html/rfc3164).
- `loggly` is a structured syslog prefix format based on RFC 5424 (https://tools.ietf.org/html/rfc5424).
- `logplex` is a Heroku-style length prefixed syslog format (https://github.com/heroku/logplex/blob/master/doc/README.http_drains.md).
- `blank` means no prefix. Just your log message. Useful when writing to JSON and CSV files.

Updating endpoints to use a different format

The following logging endpoints can be updated to use a message format other than the default:

- Amazon S3 (/guides/streaming-logs/log-streaming-amazon-s3)
- Cloud Files (/guides/streaming-logs/log-streaming-cloudfiles)
- DigitalOcean Spaces (/guides/streaming-logs/log-streaming-digitalocean-spaces)
- FTP (/guides/streaming-logs/log-streaming-ftp)
- Google Cloud Storage (/guides/streaming-logs/log-streaming-google-cloud-storage)

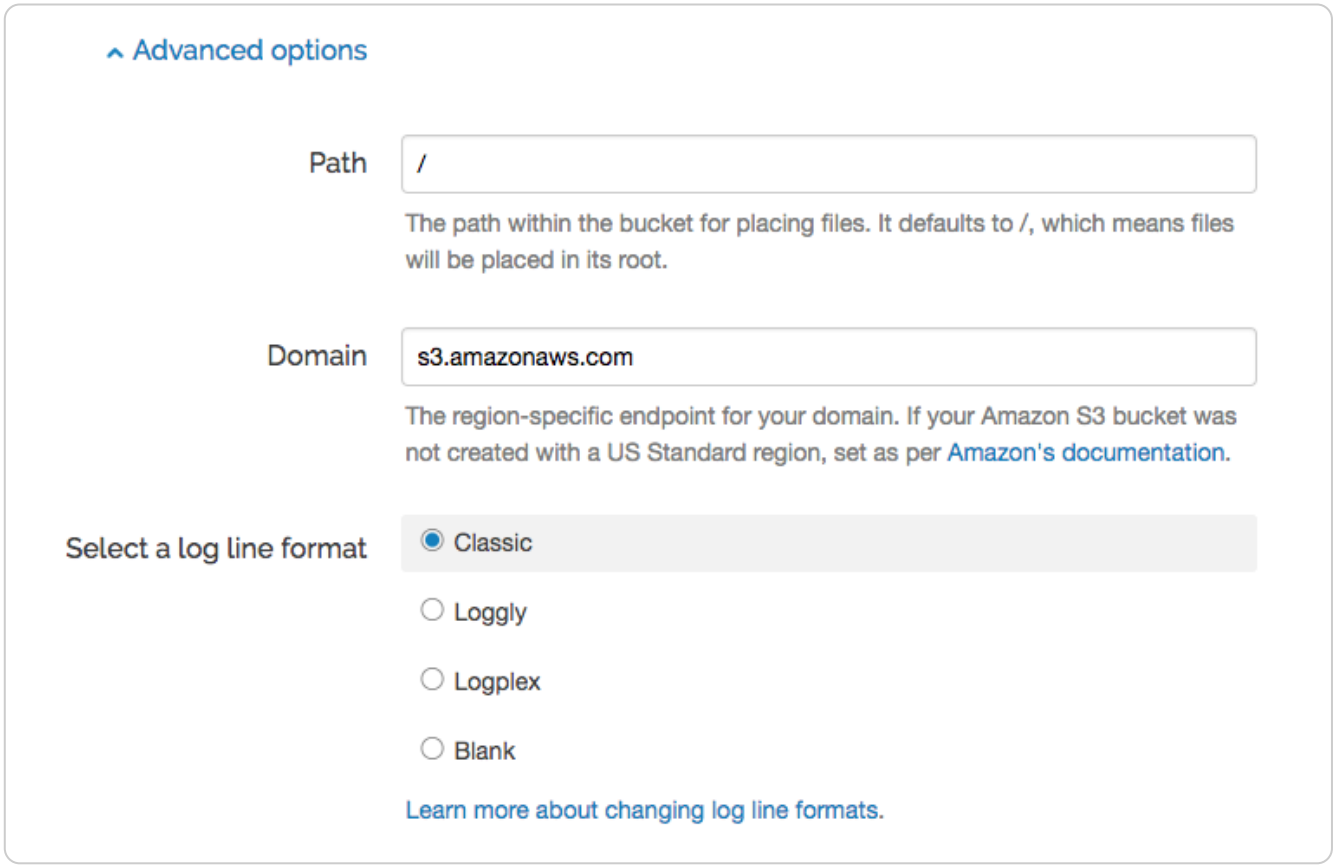
- OpenStack (/guides/streaming-logs/log-streaming-openstack)
- Sumo Logic (/guides/streaming-logs/log-streaming-sumologic)
- Syslog (/guides/streaming-logs/log-streaming-syslog)

You can use the web interface or the API to update a logging endpoint.

Using the web interface

Follow these instructions to update a logging endpoint using the web interface:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears.
5. Click the name of the logging endpoint you want to edit. The Edit this endpoint page appears.
6. Click the **Advanced options** link near the bottom of the page. The Advanced options appear.



7. In the **Select a log line format** section, select a log line format for the logging endpoint.
8. Click the **Update** button.

9. Click the **Activate** button to deploy your configuration changes.

Using the API

Run the following command to update a logging endpoint using the API:

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version number/logging/<log type>/<log name>' --data-binary '{"message_type":"<type>"}'
```

where `log type` is one of the eligible endpoint types:

- `s3`
- `cloudfiles`
- `digitalocean`
- `ftp`
- `gcs`
- `openstack`
- `sumologic`
- `syslog`

Keep in mind that the `message_type` field is a per-object field. Updating it on one logging object *will not* change it on any other objects. For example, to update a Google Cloud Storage endpoint to the `blank` message type the cURL command would look something like this if the endpoint was named "GCS Test":

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/SU1Z0isxPaozGVKXdv0eY/version/1/logging/gcs/GCS%20Test' --data-binary '{"message_type":"blank"}'
```

NOTE: The log name `GCS Test` needed to be URL encoded (https://en.wikipedia.org/wiki/Percent-encoding), which turned the space into `%20`.

§ Changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written)

For supported logging endpoints that write files to remote services, Fastly uses a combination of factors to ensure log files aren't overwritten, including:

- Using the file creation timestamp.
- Generating a unique ID.
- If a file with the same timestamp and UID combination exists, incrementing a counter and adding that to the end of the filename.

To change where log files are written, you can modify the `path` and `timestamp_format` variables. The logging system combines the `path`, `timestamp_format`, and `uid` variables to create the file name:

```
<path><timestamp>-<uid>.log<suffixes>
```

This guide explains how to use the `path` and `timestamp_format` variables to control where log files are written.

Supported logging endpoints

The following logging endpoints currently support the `path` and `timestamp_format` variables:

- Amazon S3 (/guides/streaming-logs/log-streaming-amazon-s3)
- Cloud Files (/guides/streaming-logs/log-streaming-cloudfiles)
- DigitalOcean Spaces (/guides/streaming-logs/log-streaming-digitalocean-spaces)
- FTP (/guides/streaming-logs/log-streaming-ftp)
- Google Cloud Storage (/guides/streaming-logs/log-streaming-google-cloud-storage)
- OpenStack (/guides/streaming-logs/log-streaming-openstack)
- Sumo Logic (/guides/streaming-logs/log-streaming-sumologic)
- Syslog (/guides/streaming-logs/log-streaming-syslog)

Timestamp format

You may want to consider changing the timestamp format to remove characters from the log filenames. For example, if you're working with Elastic MapReduce, you might need to remove the colons in the filename.

The `timestamp_format` variable is provided as a strftime (http://man7.org/linux/man-pages/man3/strftime.3.html) compatible format. The default format is ISO 8601 Combined Date/Time Format (https://en.wikipedia.org/wiki/ISO_8601#Combined_date_and_time_representations):

```
%Y-%m-%dT%H:%M:%S.000
```

The variables are expanded when the file is created. For example, `%Y` will be replaced by the current year and `%m` by the current month number:

`<year>-<2 digit month number>-<2 digit day number>T<hour>:<minute>:<second>`

The timestamp for a file created at midnight on January 1st, 1970 would be `1970-01-01T00:00:00.000`.

Path

The `path` variable acts differently depending on whether or not it ends in a trailing `/`.

If the variable does end in a trailing `/`, then it's treated as a directory. For example, if the variable is set to `my_logs/`, the files are written in the directory `my_logs`. If the variable is set to `my_logs` without the trailing `/`, the files are written in the top-level directory and are prefixed with `my_logs`.

The two approaches can also be combined. For example, if the variable is set to `my_logs/foo`, the files are written in the `my_logs` directory and are prefixed with `foo`.

Logs can also be nested. For example, if the variable is set to `my_logs/sub_logs/`, the files are written in the `sub_logs` directory in the `my_logs` directory.

> ★ **TIP:** The path can also be a `strftime` compatible string. For example, if the variable is set to `%Y/%m/%d`, the files are written to a directory based on the year, month, and date.

Directories are created automatically when possible.

Suffixes

Fastly's logging system automatically adds suffixes to files as appropriate.

Suffix	File type
.log	Plain log file
.log.gz	Gzipped log file
.log.gpg	PGP encrypted (/guides/streaming-logs/encrypting-logs) log file
.log.gz.gpg	PGP encrypted (/guides/streaming-logs/encrypting-logs), Gzipped log file

§ Custom log formats (/guides/streaming-logs/custom-log-formats)

Fastly provides two versions of custom log formats. All new logging endpoints use the version 2 custom log format by default. You can upgrade version 1 logging endpoints to the version 2 custom log format. You can also make version 2 look like version 1 for the sake of continuity. We've described the key advantages of the version 2 custom log format below.

Version 2 log format

This table details version 2 of Fastly's custom log formats. All variables should be prefixed by a percent sign (`%`), as indicated in the table.

Format String	Description
`%%`	The percent sign.
`%a`	The client IP address of the request.
`%A`	The local IP address.
`%B`	The size of response in bytes, excluding HTTP headers.
`%b`	The size of response in bytes, excluding HTTP headers. In Common Log Format (https://httpd.apache.org/docs/trunk/logs.html#common) (CLF), that means a `"-"` rather than a `0` when no bytes are sent.
`%{Foobar}C`	The contents of cookie `Foobar` in the request sent to the server.
`%D`	The time taken to serve the request, in microseconds.
`%{FOOBAR}e`	The contents of the environment variable `FOOBAR`. Always returns `NIL`.
`%f`	The filename.
`%h`	The remote hostname.
`%H`	The request protocol.
`%{Foobar}i`	The contents of `Foobar:` header lines in the request sent to the server.
`%I`	Bytes received, including request and headers. Cannot be zero.
`%k`	The number of keepalive requests handled on this connection. Always returns `0`.
`%l`	The remote logname (from identd, if supplied). This will return a dash unless mod_ident is present and IdentityCheck is set On. Always returns `"-"`.
`%m`	The request method.
`%{Foobar}n`	The contents of note `Foobar` from another module. Always returns `NIL`.

Format String	Description
`%{Foobar}o`	The contents of `Foobar:` header lines in the reply.
`%O`	Bytes sent, including headers. Cannot be zero.
`%p`	The canonical port of the server serving the request. Always returns `80`.
`%{format}p`	The canonical port of the server serving the request. Valid formats are `canonical`, `local`, or `remote`. Returns `80` for HTTP requests and `443` for HTTPS requests.
`%P`	The process ID of the child that serviced the request. Always returns `NIL`.
`%{format}P`	The process ID or thread ID of the child that serviced the request. Valid formats are `pid`, `tid`, and `hextid`. Always returns `NIL`.
`%q`	The query string (prepended with a `?` if a query string exists, otherwise an empty string).
`%r`	The first line of the request.
`%R`	The handler generating the response (if any).
`%s`	The status. For requests that got internally redirected, this is the status of the *original* request. Use `%>s` for the final status.
`%t`	The time the request was received, in Standard English format (e.g., `01/Jan/1970:00:00:00 -0700`). The last number indicates the timezone offset from GMT.
`%{format}t`	The time, in the form given by `format`, which should be in `strftime(3)` format (potentially localized). If the format starts with `begin:` (the default) the time is taken at the beginning of the request processing. If it starts with `end:` it is the time when the log entry gets written, close to the end of the request processing. In addition to the formats supported by `strftime(3)`, the following format tokens are supported: `sec` (number of seconds since the Epoch), `msec` (number of milliseconds since the Epoch), `usec` (number of microseconds since the Epoch), `msec_frac` (millisecond fraction), and `usec_frac` (microsecond fraction).
`%T`	The time taken to serve the request, in seconds.
`%u`	The remote user if the request was authenticated. May be bogus if return status (`%s`) is 401 (unauthorized). Always returns `"-"`.
`%U`	The URL path requested, not including any query string.

Format String	Description
`%v`	The domain name of the request. Equal to `%{req.http.host}V`.
`%V`	The same as `%v`.
`%{vcl}V`	The literal VCL to include without quoting. This can be used to write VCL variables to your logs (e.g., `%{client.geo.country_code}V` or `%{tls.client.cipher}V`). This %-directive is a Fastly extension and is not found in Apache.
`%X`	The connection status when response is completed. Statuses include `X` (connection aborted before the response completed), `+` (connection may be kept alive after the response is sent), and `-` (connection will be closed after the response is sent).

Version 1 log format

This table details version 1 of Fastly's custom log formats. All variables should be prefixed by a percent sign (`%`), as indicated in the table.

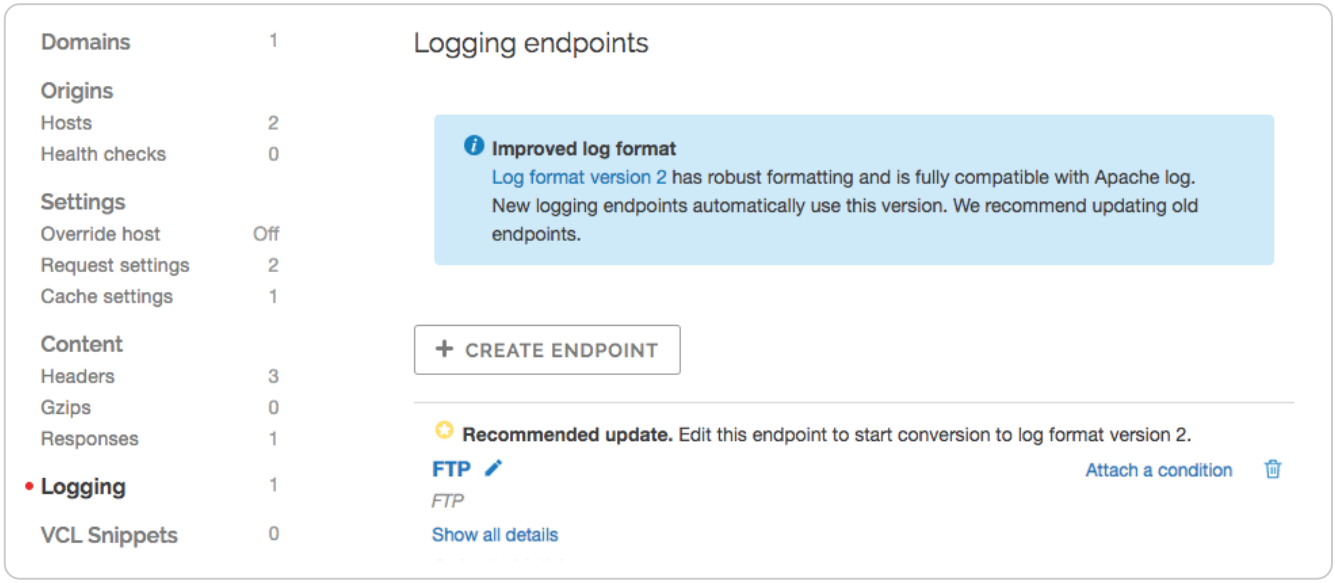
Format String	Description
`%b`	The content size of the response, calculated using the `Content-Length` header rather than actually checking the length of the response (and may therefore be wrong).
`%h`	The remote IP address.
`%l`	The remote log name. Always returns the hardcoded value `"-"`.
`%r`	The HTTP verb and request path (e.g., `GET /index.html`). Unlike Apache and version 2 log formats, the protocol version is not included.
`%>s`	The status of the last request.
`%t`	The time the request was received, in Unix `ctime` format (e.g., `Thu, 01 Jan 1970 00:00:00 GMT`) rather than Apache's Standard English format (e.g., `01/Jan/1970:00:00:00 -0700`).
`%u`	The remote user. Always returns the hardcoded value `"-"`.

Upgrading endpoints to use version 2 log format

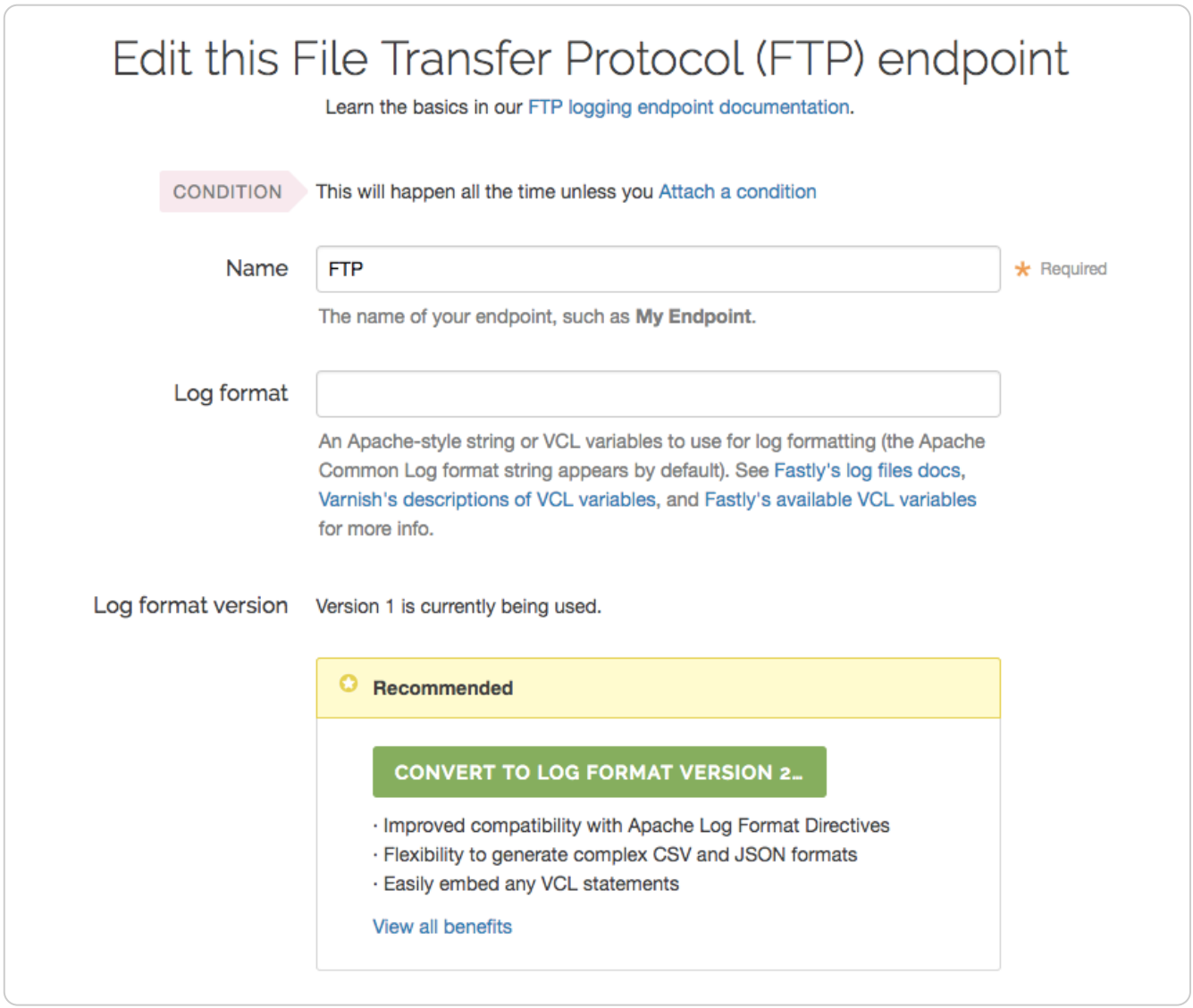
⚠ **WARNING:** Upgrading is a permanent change. Logging objects using version 2 formatting cannot be downgraded to version 1.

Follow these instructions to upgrade a logging endpoint to the version 2 custom log format:

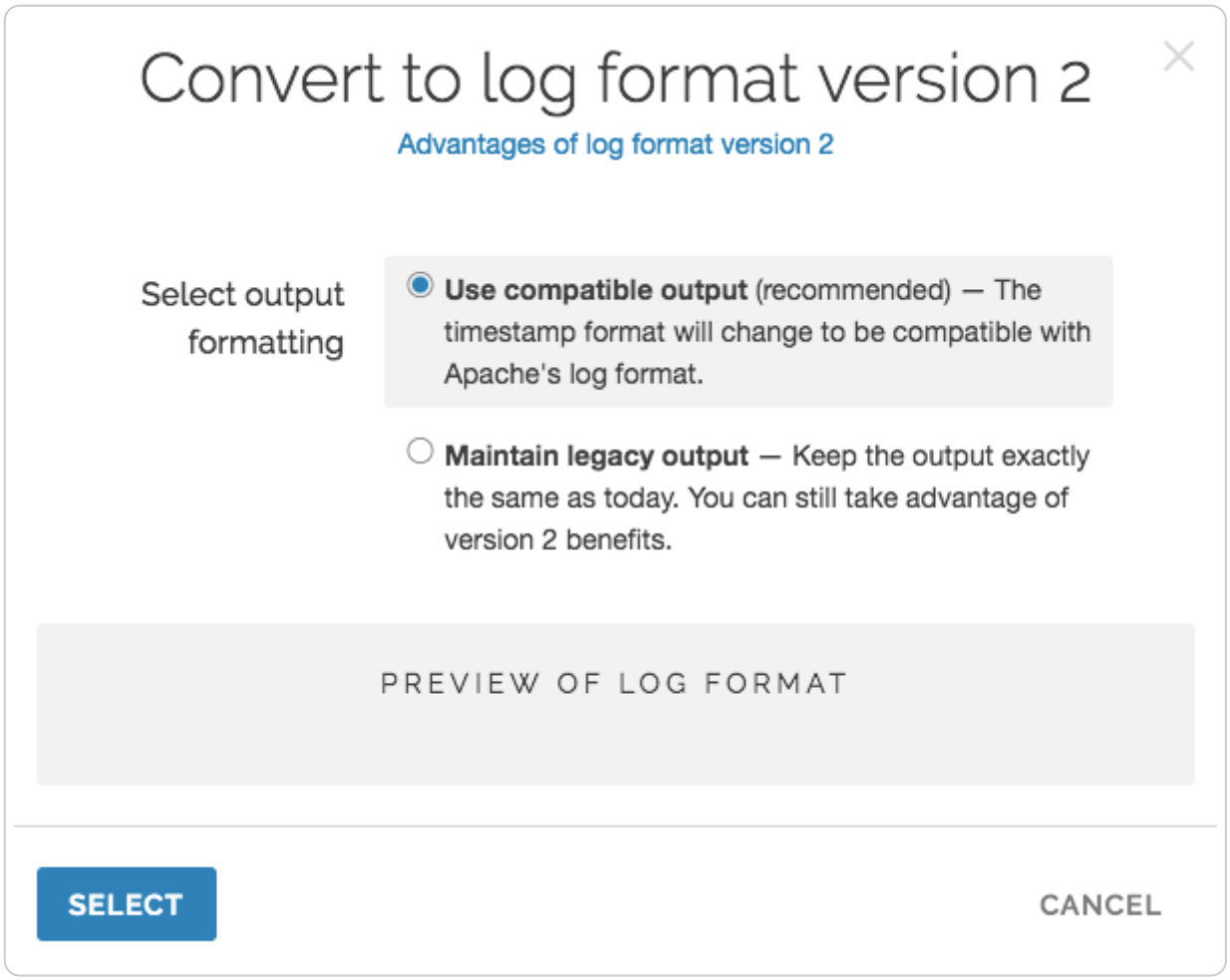
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears. If you have any logging endpoints using the version 1 custom log format, a message appears indicating that they can be updated.



5. Click the name of a logging endpoint to edit it. The Edit this endpoint page appears.



6. Click the **Convert to Log Format Version 2** button. The Convert to log format version 2 window appears.



7. Select an output format:
   - **Use compatible output** is the recommended setting. This setting won't modify your timestamp format string, but your logs will be formatted differently. The new format will be compatible with Apache's log format.
   - **Maintain legacy output** uses the version 2 parser, but the generated log string will be the same. This means that any instances of `%t` need to be turned into `%{now}V`, any instances of `%r` need to be turned into `%{req.url}V`, and any instances of `%b` need to be turned into `%{resp.http.Content-Length}V`.
8. Click the **Select** button. The Edit this endpoint page appears.
9. Click the **Update** button to upgrade the logging endpoint to the version 2 custom log format.
10. Click the **Activate** button to deploy your configuration changes.

Using the API to upgrade

To upgrade a logging endpoint using the Fastly API (/api), either clone the active version of the service you need upgraded or choose a version of the service that is unlocked and not active, then run the following command on that in development version:

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version number>/logging/<log type>/<log name>' --data-binary '{"format_version":"2"}'
```

where `log type` is type of the endpoint you want to upgrade:

- `ftp`
- `heroku`
- `logentries`
- `loggly`
- `logshuttle`
- `papertrail`
- `s3`
- `scalyr`
- `sumologic`
- `syslog`

Keep in mind that the `format_version` field is a per-object field. Updating it on one logging object will *not* change it on any other objects. For example, to upgrade a Google Cloud Storage endpoint the cURL command would look something like this if the endpoint was named "GCS Test":

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/SU1Z0isxPaozGVKXdv0eY/version/1/logging/gcs/GCS%20Test' --data-binary '{"format_version":"2"}'
```

NOTE: The log name `GCS Test` needed to be URL encoded (https://en.wikipedia.org/wiki/Percent-encoding), which turned the space into `%20`.

Determining which logging version is being used

To determine which logging version your service currently uses, issue the following cURL command:

```
curl -X GET -H 'Fastly-Key: FASTLY_API_TOKEN'  'https://api.fastly.com/service/<your Fastly service ID>/version/<version number>/logging/<log type>/<log name>'
```

where `<log type>` is `ftp`, `heroku`, `logentries`, `loggly`, `logshuttle`, `papertrail`, `s3`, `scalyr`, `sumologic`, or `syslog`. The cURL command will produce JSON output detailing the configuration of your service version. For example:


```
{
    "address": "logs.papertrailapp.com",
    "created_at": "2016-04-01T15:37:30+00:00",
    "deleted_at": null,
    "format": "time.start.msec time.to_first_byte time.elapsed req.body_bytes_read req.bytes_re
ad resp.http.content-length server.region client.ip %>s \"req.request req.url req.proto\" \"re
q.http.referer\" \"req.http.user-agent\"",
    "format_version": "2",
    "hostname": "logs.papertrailapp.com",
    "name": "fastly",
    "port": "11111",
    "public_key": null,
    "response_condition": "LOG /",
    "service_id": "1a2b3c4d5e6f7g8h9j0k",
    "updated_at": "2016-04-01T19:47:47+00:00",
    "version": "123"
}
```


The `format_version` field displays either a `1` or a `2` as appropriate for the custom log format being used.

Advantages of using the version 2 custom log format

The key advantages of using the version 2 custom log format include the following:

- Log lines are generated in `vcl_log` instead of `vcl_deliver` to allow us to accurately set the various size variables because `vcl_log` is run after the object has been delivered to the browser.
- The `%t` time directive is compatible with Apache log format. In version 1, we used a non-standard time format.
- The `%r` "first line of request" directive is compatible with Apache log format. In version 1, we incorrectly left off the protocol.
- When using the `%b` directive, which represents the size of a response in bytes, excluding HTTP headers is more accurate. In version 1, we used the reported Content-Length from the origin, which could be inaccurate (especially with ESI (/guides/performance-tuning/using-edge-side-includes)).
- We've added all Apache logging directives that make sense. In version 1, we used a smaller subset.

Making version 2 logs look like version 1

The default logging format for version 1 is as follows:

```
%h %l %u %t %r %>s
```

Most of the directives in version 2 are exactly the same - only `%t` and `%r` are different. After you upgrade to version 2 log formats, you can recreate the appearance of version 1 logs using the new `%{...}V` directive, which allows you to specifically include VCL in logging directives:

```
%h %l %u %{now}V %{req.request}V %{req.url}V %>s
```

In addition, if you are using the `%b` directive in version 1, then you can use this directive instead:

```
%{resp.http.Content-Length}V
```

---

§ Encrypting logs (/guides/streaming-logs/encrypting-logs)

For supported logging endpoints, Fastly allows you to encrypt your log files before they are written to disk. The files are encrypted using OpenPGP (Pretty Good Privacy) (https://en.wikipedia.org/wiki/Pretty_Good_Privacy).

Supported logging endpoints

The following logging endpoints currently support PGP encryption:

- Amazon S3 (/guides/streaming-logs/log-streaming-amazon-s3)
- Cloud Files (/guides/streaming-logs/log-streaming-cloudfiles)
- DigitalOcean Spaces (/guides/streaming-logs/log-streaming-digitalocean-spaces)
- FTP (/guides/streaming-logs/log-streaming-ftp)
- Google Cloud Storage (/guides/streaming-logs/log-streaming-google-cloud-storage)
- OpenStack (/guides/streaming-logs/log-streaming-openstack)

Generating a PGP key pair

To use this feature, you'll need to use a PGP implementation (such as GPG (https://gnupg.org)) to generate a public and private PGP key pair. Typically, this involves running the following command in a terminal application on your personal computer:

```
gpg --gen-key
```

Follow the instructions shown in your terminal application. Enter your email address and set a passphrase when prompted. Remember the values you enter.

> ⚠ **WARNING:** Keep your private key safe! If you lose it, your encrypted log files will be permanently unreadable.

Exporting the PGP public key

After you generate the PGP key pair, you'll need to export your public key. Typically, this involves running the following command in a terminal application on your personal computer:

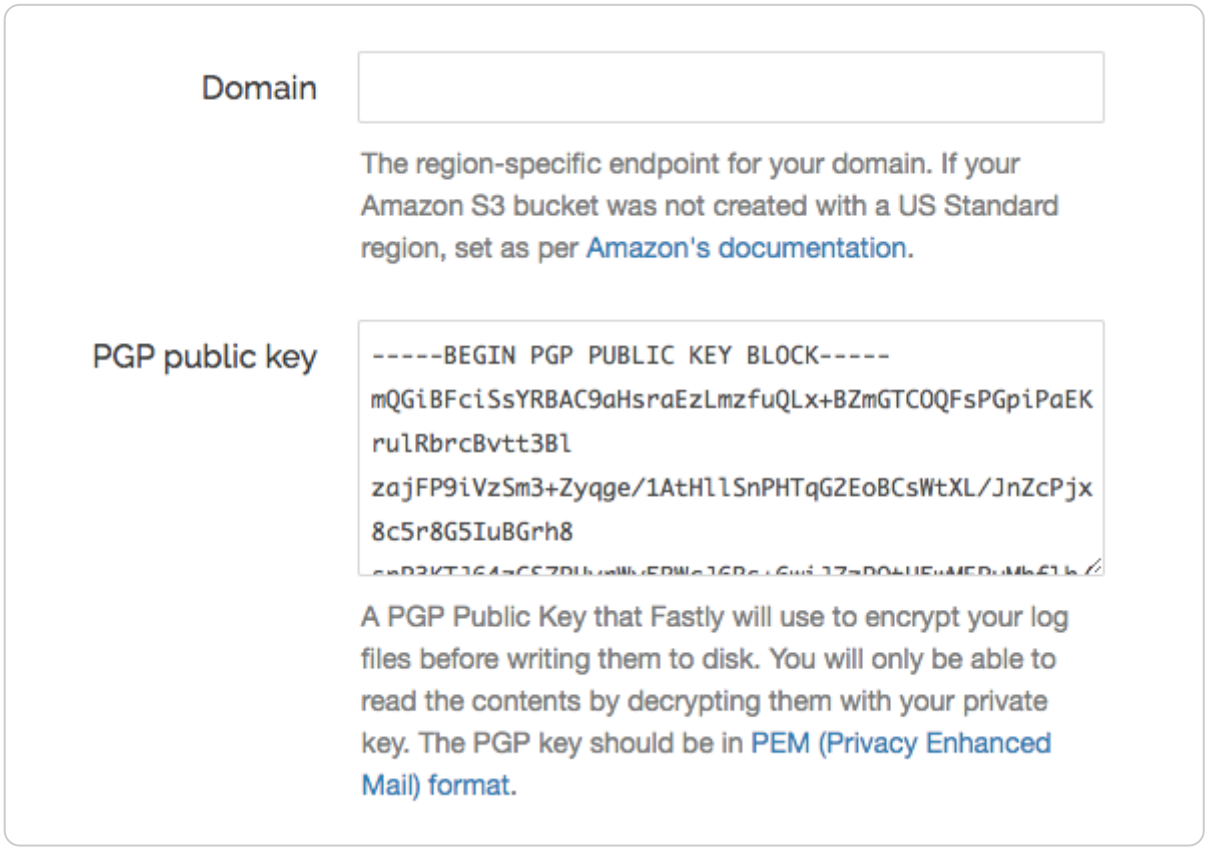
```
gpg --armor --export <your email>
```

The output will be in PEM (Privacy-Enhanced Mail) (https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail) format and will look similar to the following:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQGiBFciSsYRBAC9aHsraEzLmzfuQLx+BZmGTCOQFsPGpiPaEKrulRbrcBvtt3Bl
zajFP9iVzSm3+Zyqge/1AtHllSnPHTqG2EoBCsWtXL/JnZcPjx8c5r8G5IuBGrh8
snP3KTJ64zCS7PUvrWy5RWcJ6Rs+6wiJ7zPOtU5wMEPuMbflh/soy50zrwCg74XN
u/jQYfGKTLTtap+hNPhO1o0P/2+Bqj7o3CgEkQQ8RRF+iVFPgt/5HEXpS6TxjJPJ
FFv2t311rwlJgPgH4nOuRwXRkJ+woPmZE2UVsG9bmV1296fq7o2HiPWUUHafUlPV
9ib4xlu2MPkCmoKVzOBKBxZ5kXYAYVchJnERrI8sPOATY+UG2zJyomPnUN3mOC7L
z1fcA/9aQaSOcPxxTJfT+JOuMwQuNbFJvrIR/QqEam4x/6hRlnLatVb7wRlKa5o2
9Pn7eGVLWIo791zwEALQSpvXIYasjVrJNPVGyORIMhMrgP0HUX3oKTX+iO7AepcS
8jAsLjTYNP/sP+n6/YAGQ2JckSBA/28s63K/Ud+NQE8EGBECAA8FAlcQSsYCGwwF
Cq+C9z2XwKzHLPyFIy5Fz1QIvH9iyZQkn8WbIExXojvE5WzfEbQfU2ltb24gV2lz
bHLAjtCG4qYNhebb4efLxyzW4b23WMC+SQa+3QKDa3PYONQDfsxzR2GvYJQLVWTu
CAcPAgQVAggPBBYCAwECHgECF4AACgkQjWBU6PMWoWUjUwCfUYPbGB+2CIhWbnWb
7zYmQKDIYTUAoKq5uAkfAmTp6CSkYw0l9C0vgpc4uQINBFciSsYQCADrWqWLv6TO
/JmDVPOJjmpve65/e9wGrgh4h5MotCVe7r+b4tplpkwC1poC97c6W4/Z2QKe2Wco
u+D5tEPugFf0Garn189P97L6tNRXJbR5VbDNrulyR18gJN0Bq4Du+/1kkmF2QRLQ
xCLhmE0S4HvSCLQY6FQHetDOn0jEP6covaX7U+ksXjpyASczaxPWA8Cnk9KrQ/mC
wAzx2CzgMX3FFwwEG43wlm1u+QnfuV6QOlQtFtclE/8eJ4WWdpAHkLwZDeEyrgvh
5zA5YhCnt2NbdwF32r6QOLhLL3sge/rsZ09ten+NBM6HJWDYTFkRlyOkVQTyNEsT
GUEgx3Q7/1c3AAMFB/4g5AhsoLRxbFGKJFbDEQxRYW1QQH6ChNMTqYA0k4vJ+Cbv
dG93QDxzaW1vbkBmYXN0bHkuY29tPohmBBMRAgAmBQJXQkrGAhsDBQkAG6+ABgsJ
41QqfQTWunwbCqAQPhuxYhUqaSBV9Wb1NUnwLzv0fkfJLTBp5X5B1eQGBdFFcpFa
+Rz79gfn41sylQlgRx1Xv79M5PObyAPuJDxBaSQ/lVzBfsK8lxxr90VvnJy4jpSe
QQgqFSQjxVxpdAd3gt0RaU9Tds0dkRy+kJNh31UJQShhFYBP58Q9Qr9A6NWPmJ2b
CQAbr4AACgkQjWBU6DMWoWVkAwCfX50rK3UXVQKz+F+r5qv2czQ9hcQAn3wQIgZ+
Mlw8D2qcp71wCZmX/mxz
=MiF8
-----END PGP PUBLIC KEY BLOCK-----
```

Enabling log encryption

To enable PGP encryption for a logging endpoint, copy and paste your public PGP key into the Fastly web interface. Follow the instructions in the logging endpoint guides.



Decrypting log files

To read an encrypted log file, you'll need to download and decrypt it. Typically, this involves running the following command in a terminal application on your personal computer:

```
gpg --decrypt <encrypted log file>
```

Enter your passphrase to decrypt the log file.

---

§ Log streaming: Amazon S3 (/guides/streaming-logs/log-streaming-amazon-s3)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to Amazon Simple Storage Service (https://aws.amazon.com/s3/) (Amazon S3). Amazon S3 is a static file storage service used by developers and IT teams. You can also use the instructions in this guide to configure log streaming to another S3-compatible service.

NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

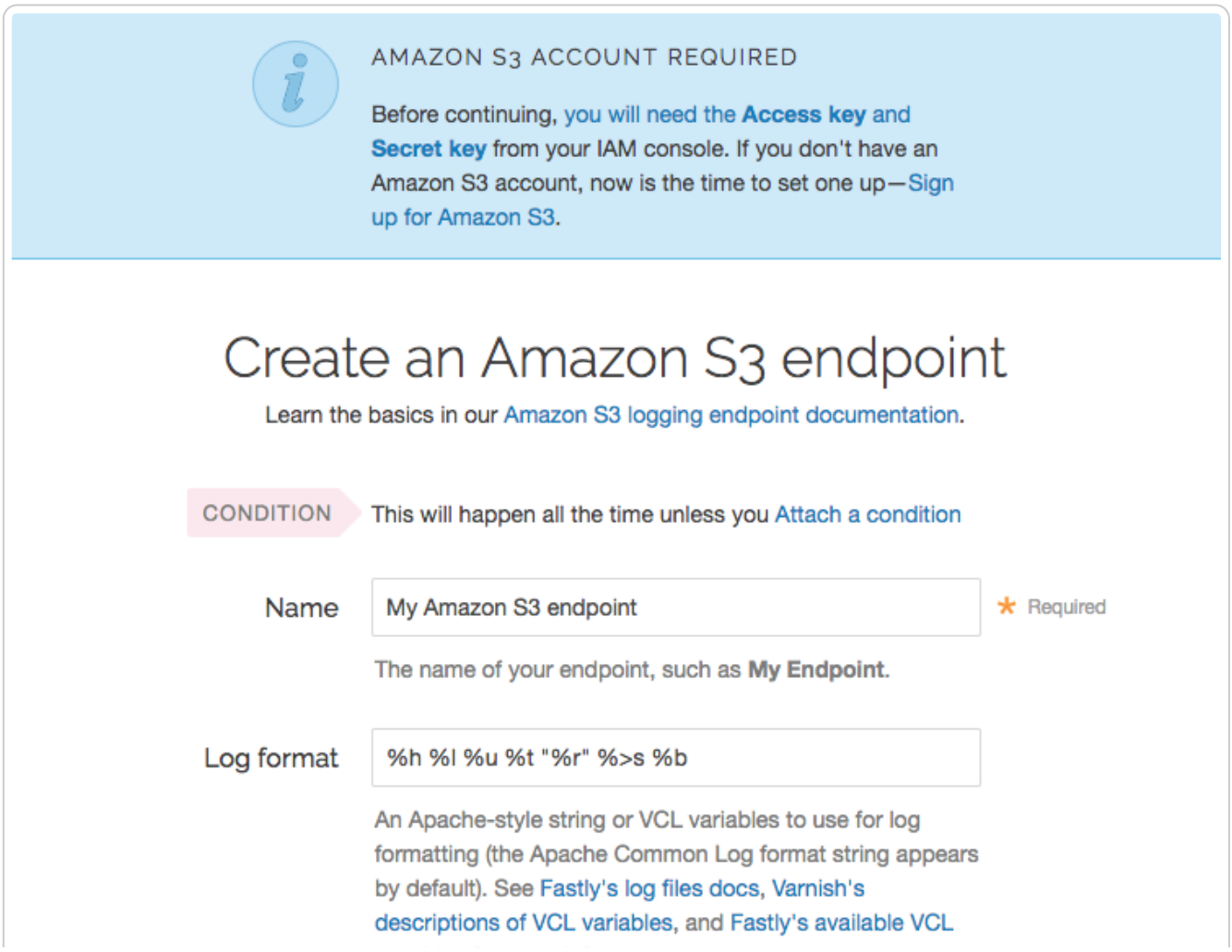
Prerequisites

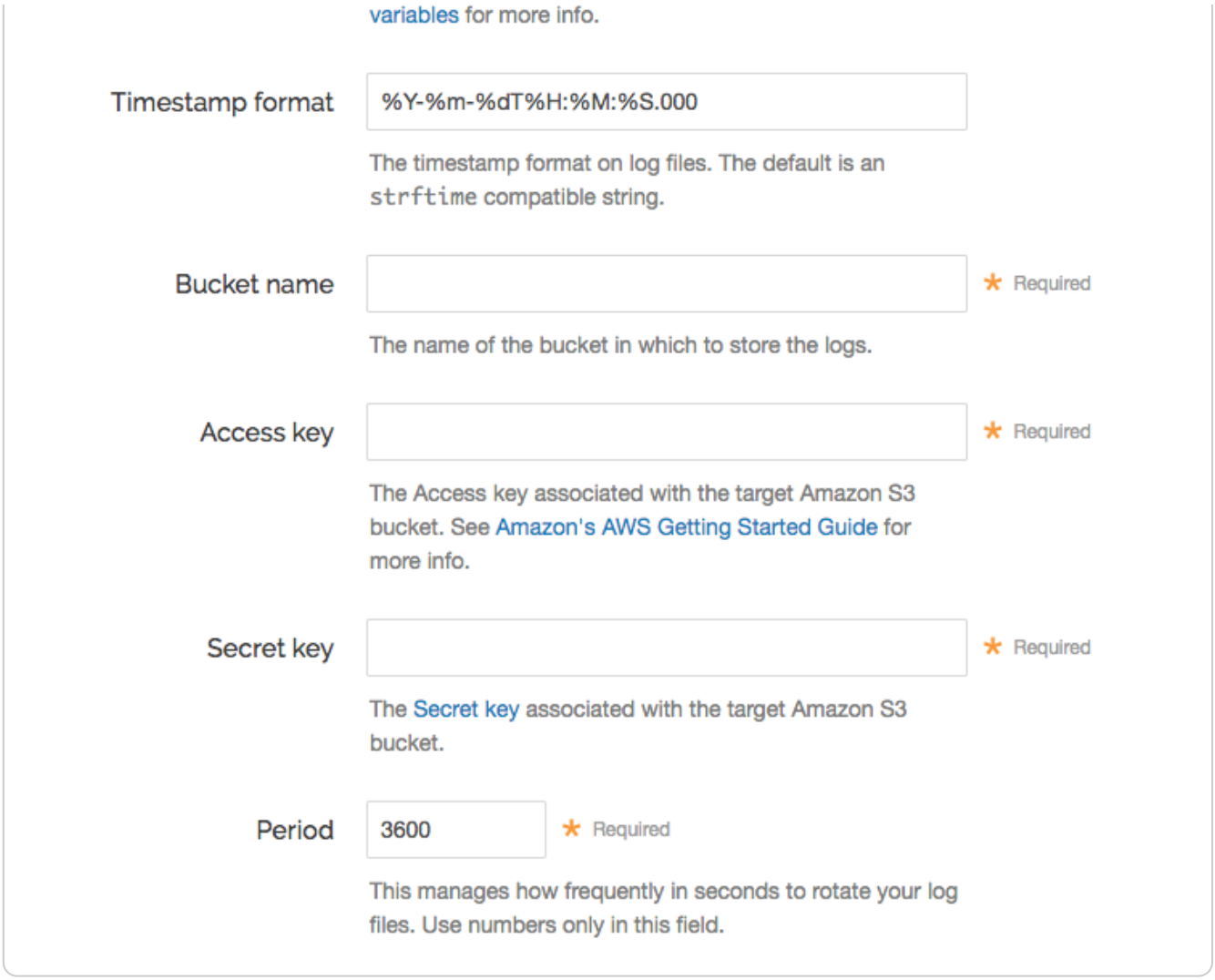
Before adding Amazon S3 as a logging endpoint for Fastly services, we recommend creating an Identity and Access Management (IAM) user in Amazon S3 specifically for Fastly. Grant the user `ListBucket`, `GetObject`, and `PutObject` permissions for the directory in which you want to store logs. For more information, see Amazon's Getting Your Access Key ID and Secret Access Key (http://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html) page.

Adding Amazon S3 as a logging endpoint

After you've registered for an Amazon S3 account and created an IAM user in Amazon S3, follow these instructions to add Amazon S3 as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Amazon Web Services S3 logo. The Create an Amazon S3 endpoint page appears.



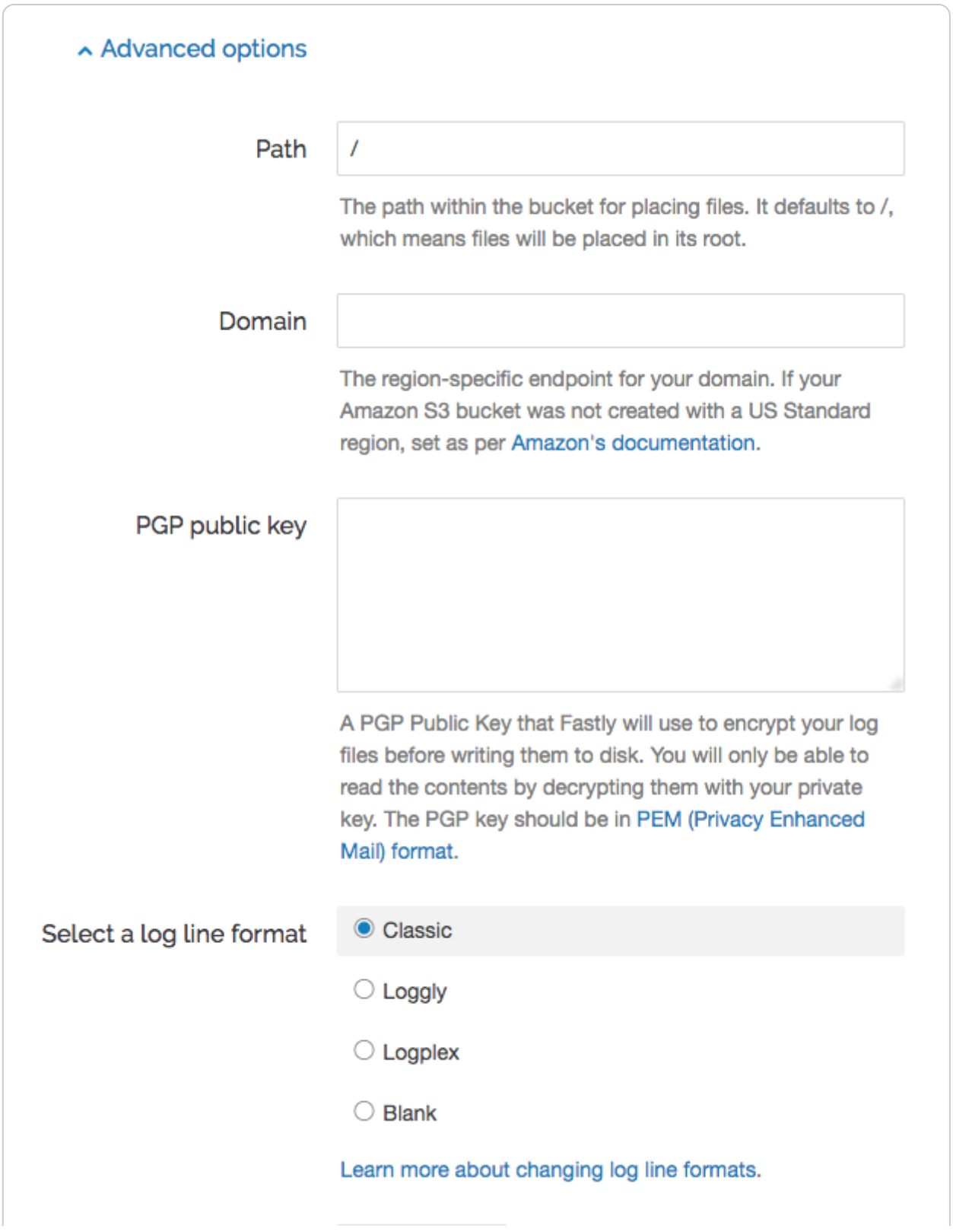


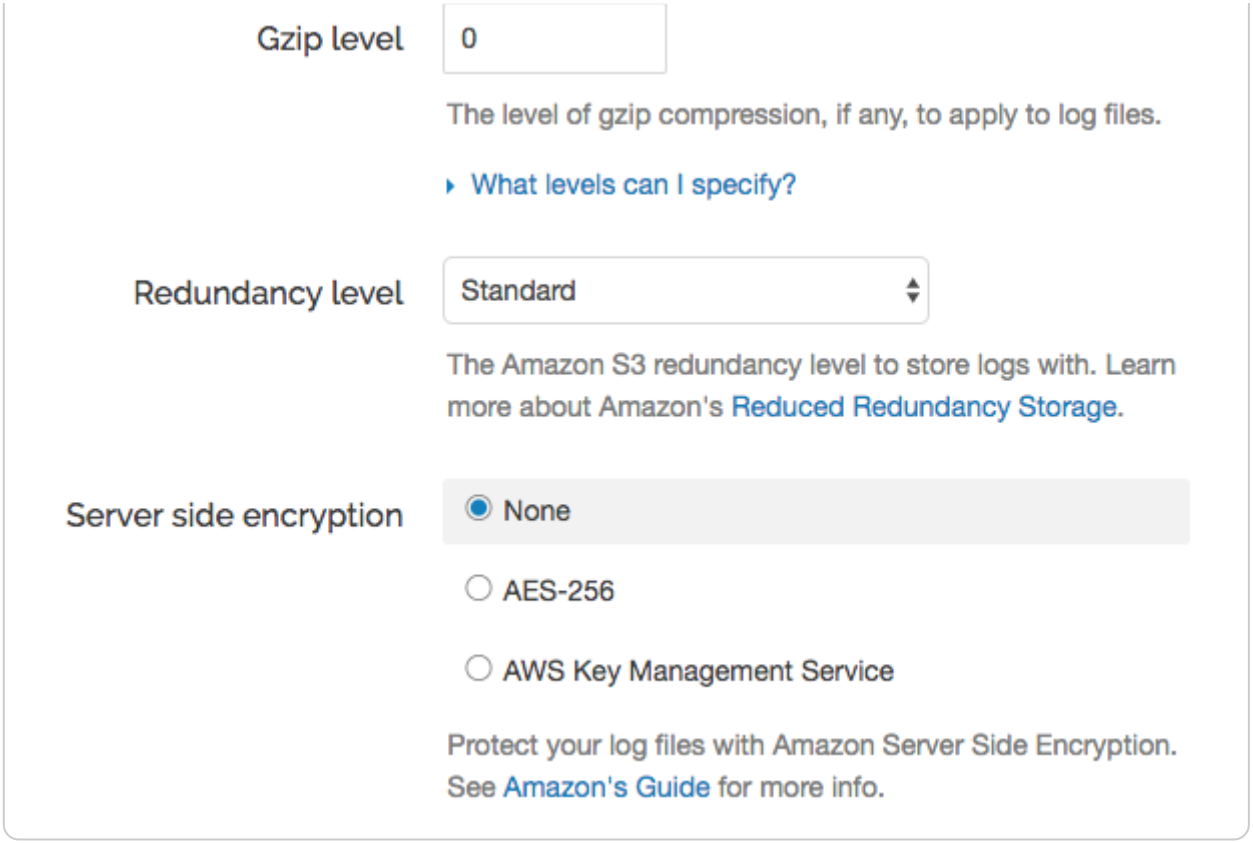
3. Fill out the **Create an Amazon S3 endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
   - In the **Timestamp format** field, optionally type a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **Bucket name** field, type the name of the Amazon S3 bucket in which to store the logs.
   - In the **Access key** field, type the access key associated with the Amazon S3 bucket. See Amazon's S3 Getting Started Guide (http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/A for more information.
   - In the **Secret key** field, type the secret key associated with the Amazon S3 bucket. See Amazon's S3 Getting Started Guide

(http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/A for more information.

- In the **Period** field, optionally type an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a new S3 endpoint** page and decide which of the optional fields to change, if any.





5. Fill out the **Advanced options** of the **Create an Amazon S3 endpoint** page as follows:
   - In the **Path** field, optionally type the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **Domain** field, optionally type the domain of the Amazon S3 endpoint. If your Amazon S3 bucket was not created in the US Standard region, you must set the domain to match the appropriate endpoint URL. Use the table in the S3 section of the Regions and Endpoints (http://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region) Amazon S3 documentation page. If you want to use an S3-compatible storage system (such as Dreamhost's DreamObjects (https://www.dreamhost.com/cloud/storage/)), set the domain to match the domain name for that service (for example, in the case of DreamObjects, the domain name would be `objects.dreamhost.com`).
   - In the **PGP public key** field, optionally type a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format (https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail). See our guide on log encryption (/guides/streaming-logs/encrypting-logs) for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats)

provides more information.

- In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
- From the **Redundancy level** menu, select a setting. This value defaults to **Standard**. Amazon's Using Reduced Redundancy Storage Guide (http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingRRS.html) provides more information on using reduced redundancy storage.
- In the **Server side encryption** area, optionally select an encryption method to protect files that Fastly writes to your Amazon S3 bucket. Valid values are **None**, **AES-256**, and **AWS Key Management Service**. If you select **AWS Key Management Service**, you'll have to provide an AWS KMS Key ID. See Amazon's guide on protecting data using server-side encryption (http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html) for more information. Our discussion of format strings (/guides/streaming-logs/custom-log-formats) also provides more information.

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

NOTE: Although Fastly continuously streams logs into Amazon S3, the Amazon S3 website and API do not make files available for access until after their upload is complete.

§ Log streaming: Cloud Files (/guides/streaming-logs/log-streaming-cloudfiles)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log file to Cloud Files (https://www.rackspace.com/cloud/files). Operated by Rackspace, Cloud Files is a file storage service used by developers and IT teams.

NOTE: This logging endpoint is disabled by default. To enable this endpoint for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Prerequisites

If you don't already have a Rackspace Cloud account, you'll need to register (https://cart.rackspace.com/cloud) for one. Follow the instructions on Rackspace's website (https://cart.rackspace.com/cloud).

Creating a Cloud Files user and container

Start by creating a Cloud Files user with restricted permissions via Rackspace's cloud control panel (https://mycloud.rackspace.com/).

1. Log in to Rackspace's cloud control panel (https://mycloud.rackspace.com/).
2. From the user account menu, select **User Management**.
3. Click **Create User** and fill in all appropriate details.
4. In the **Product Access** section, set **User Role** to **Custom**.
5. Review the **Product Access** list. For all items in the **Product** column, set **Role** to **No Access** except the **Files** item.
6. Set the **Files** item **Role** to **Admin**. This will allow you to create the files to store the logs in, but not access any other services.

Next, find the API key for your Cloud Files account. You'll use this later to authenticate using the Cloud Files API.

1. From the user account menu, select **Account Settings**.
2. Show the API key in the **Login details** and make a note of it.

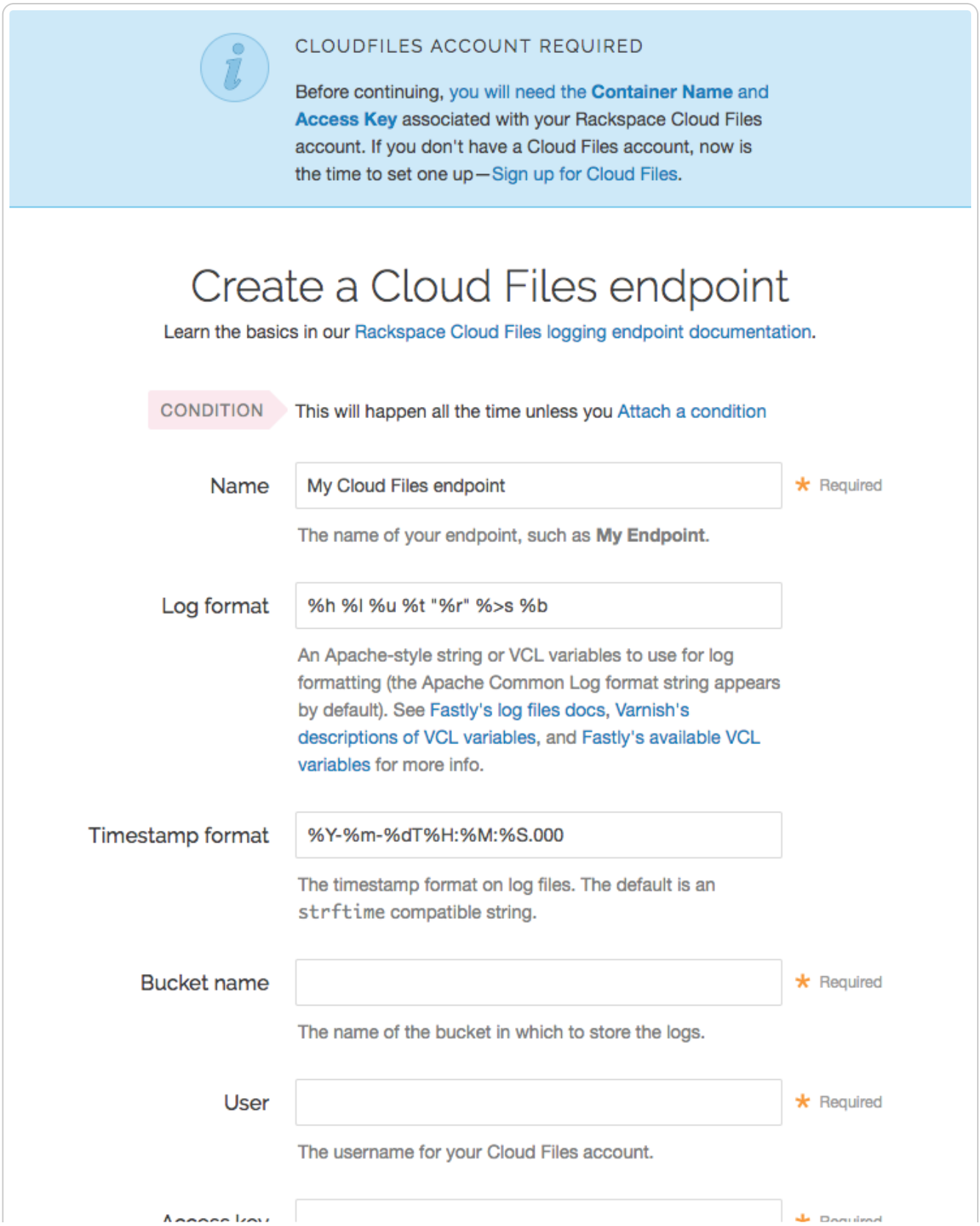
Now that you've created the Cloud Files user and found the API key, you can set up a Cloud Files container.

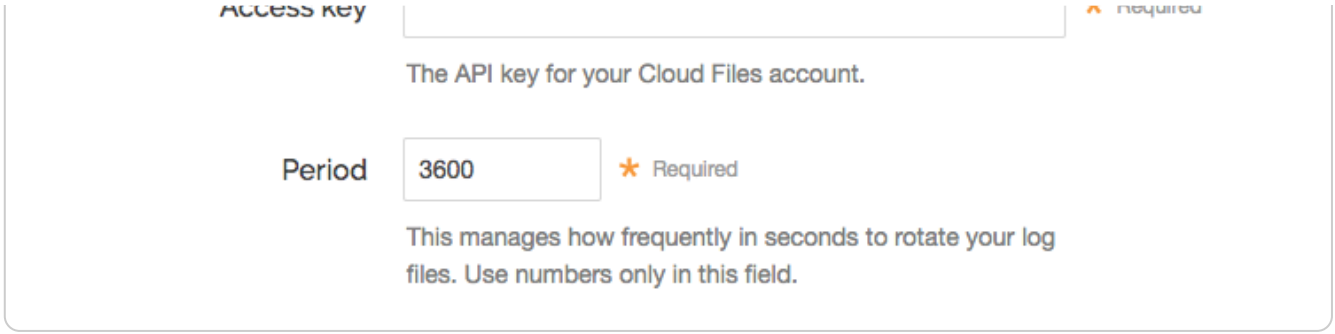
1. From the **Storage** menu, select **Files**.
2. Click **Create Container**.
3. Assign the container a meaningful name like `Fastly logs - my service`.
4. Choose a region to keep the files in and make sure the container is private.
5. Click **Create Container**.

Adding a Cloud Files logging endpoint

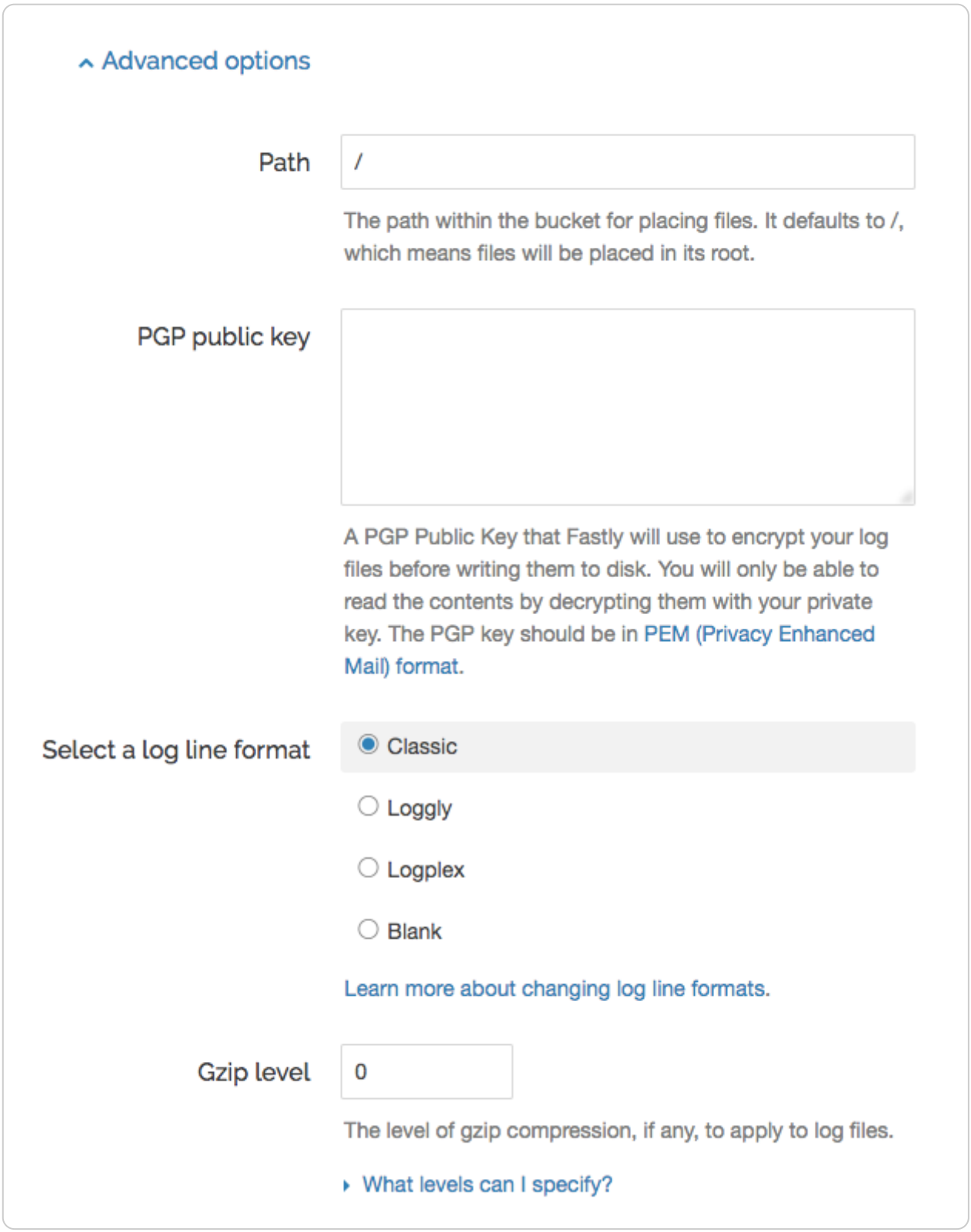
Once you have created the Cloud Files user and container, follow these instructions to add Cloud Files as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Cloud Files logo. The Create a Cloud Files endpoint page appears.





3. Fill out the **Create a Cloud Files endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings (/guides/streaming-logs/custom-log-formats) for more information.
   - In the **Timestamp format** field, optionally type a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **Bucket name** field, type the name of the Cloud Files container in which to store the logs.
   - In the **User** field, type the username of the Cloud Files user you created above.
   - In the **Access key** field, type the API key of your Cloud Files account.
   - In the **Period** field, type an interval (in seconds) to manage how frequently in seconds to rotate your log files. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a Cloud Files endpoint** page and decide which of the optional fields to change, if any.



5. Fill out the **Advanced options** of the **Create a Cloud Files endpoint** page as follows:
   - In the **Path** field, optionally type the path within the container to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the container's root path. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.

- In the **PGP public key** field, optionally type a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format (https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail). See our guide on log encryption (/guides/streaming-logs/encrypting-logs) for more information.
- In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats) provides more information.
- In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

§ Log streaming: DigitalOcean Spaces (/guides/streaming-logs/log-streaming-digitalocean-spaces)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to DigitalOcean Spaces (https://www.digitalocean.com/products/storage/object-storage/). DigitalOcean Spaces is an Amazon S3-compatible static file storage service used by developers and IT teams.

❗ **IMPORTANT:** This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

ℹ **NOTE:** This logging endpoint is disabled by default. To enable this endpoint for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

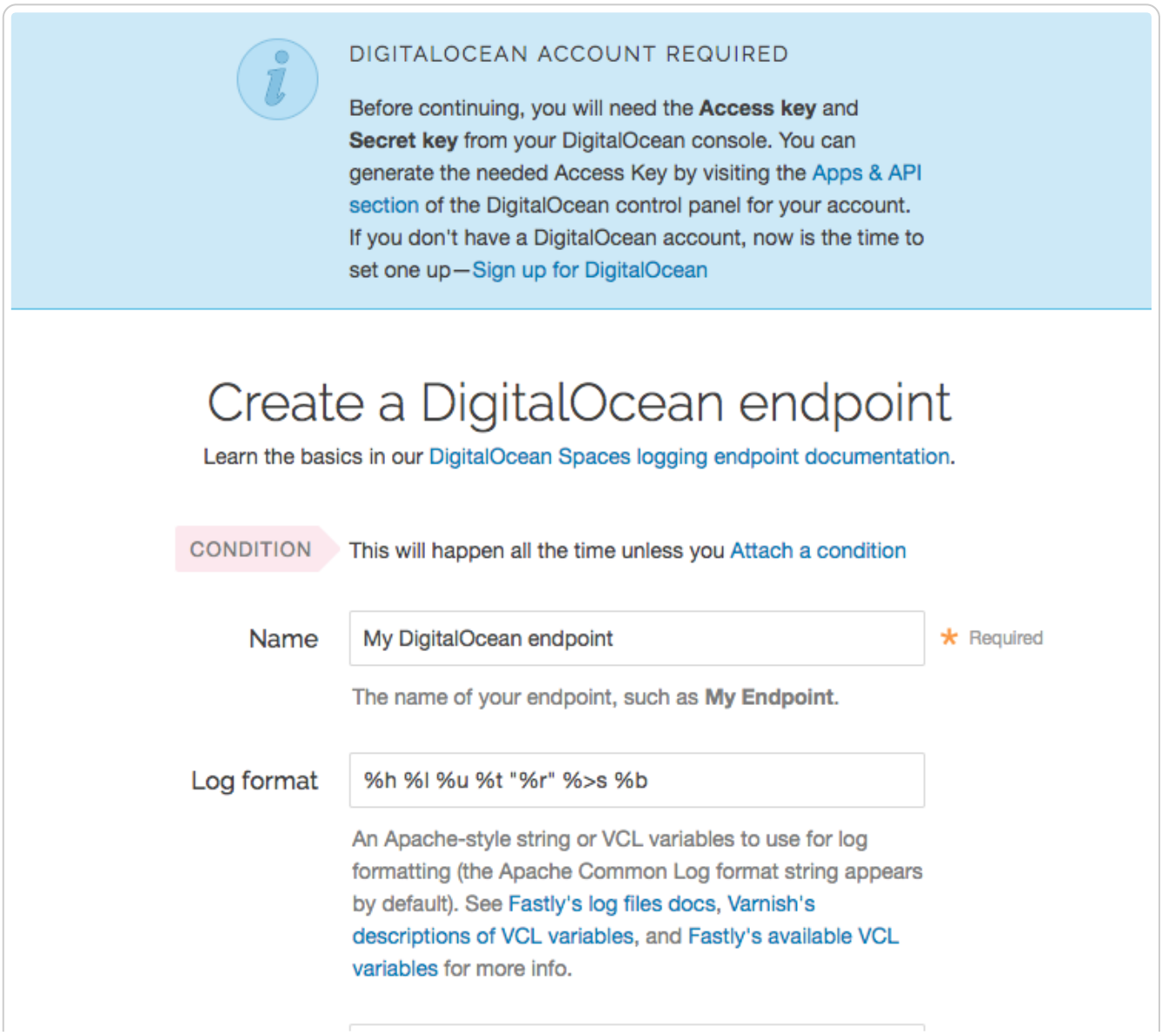
Prerequisites

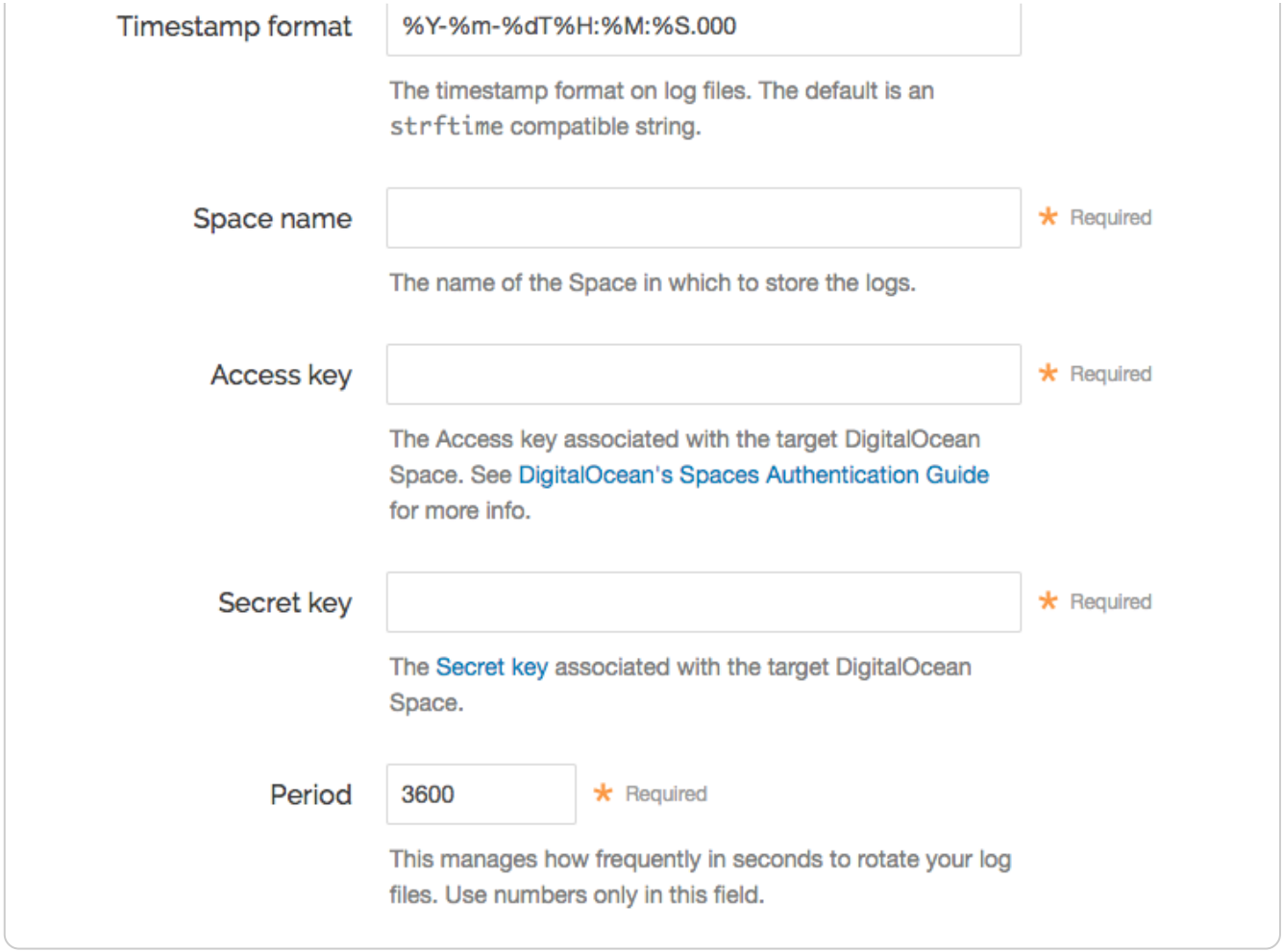
Before adding DigitalOcean Spaces as a logging endpoint for Fastly services, you'll need to create a DigitalOcean account (https://www.digitalocean.com/) if you don't already have one. Then you'll need to create a space with private access permissions on DigitalOcean's website, generate a secret key and an access key (https://developers.digitalocean.com/documentation/spaces/#authentication), and make a note of the endpoint.

Adding DigitalOcean Spaces as a logging endpoint

After you've created a DigitalOcean Space, follow these instructions to add DigitalOcean Spaces as a logging endpoint:

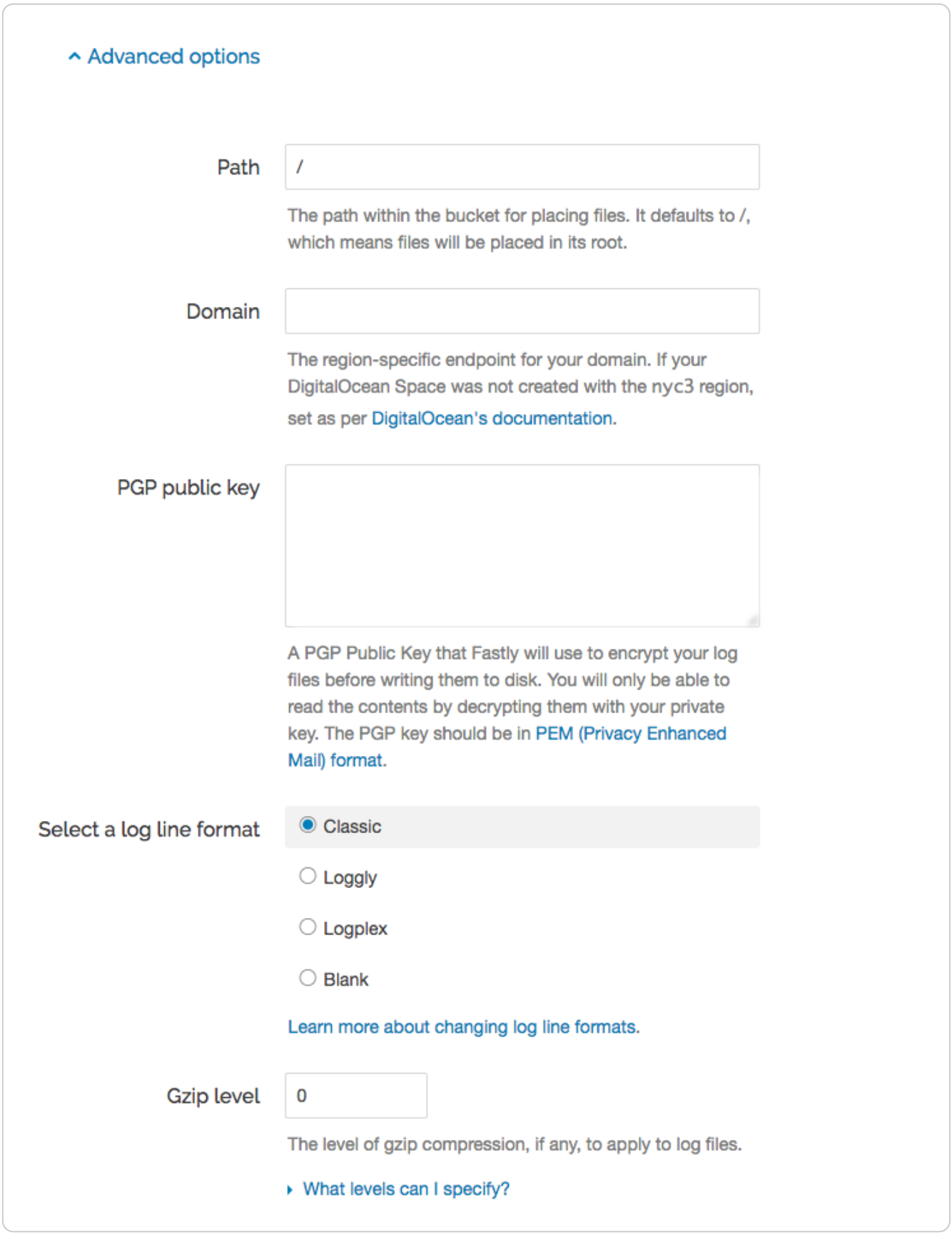
1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the DigitalOcean Spaces logo. The Create a DigitalOcean endpoint page appears.





3. Fill out the **Create a DigitalOcean endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
   - In the **Timestamp format** field, optionally type a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **Bucket name** field, type the name of the DigitalOcean Space in which to store the logs.
   - In the **Access key** field, type the access key associated with the DigitalOcean Space. See the DigitalOcean Spaces Authentication Guide (https://developers.digitalocean.com/documentation/spaces/#authentication) for more information.
   - In the **Secret key** field, type the secret key associated with the DigitalOcean Space.
   - In the **Period** field, optionally type an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a DigitalOcean Endpoint** page and decide which of the optional fields to change, if any.



5. Fill out the **Advanced options** of the **Create a DigitalOcean endpoint** page as follows:

- In the **Path** field, optionally type the path within the container to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the container's root path. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
- In the **Domain** field, type the region-specific endpoint for your domain. In most cases, this should be `nyc3.digitaloceanspaces.com`. If the DigitalOcean Space was not created in the `nyc3` region, refer to DigitalOcean's documentation (https://developers.digitalocean.com/documentation/spaces/#introduction) to find the correct domain.
- In the **PGP public key** field, optionally type a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format (https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail). See our guide on log encryption (/guides/streaming-logs/encrypting-logs) for more information.
- In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats) provides more information.
- In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

---

§ Log streaming: FTP (/guides/streaming-logs/log-streaming-ftp)

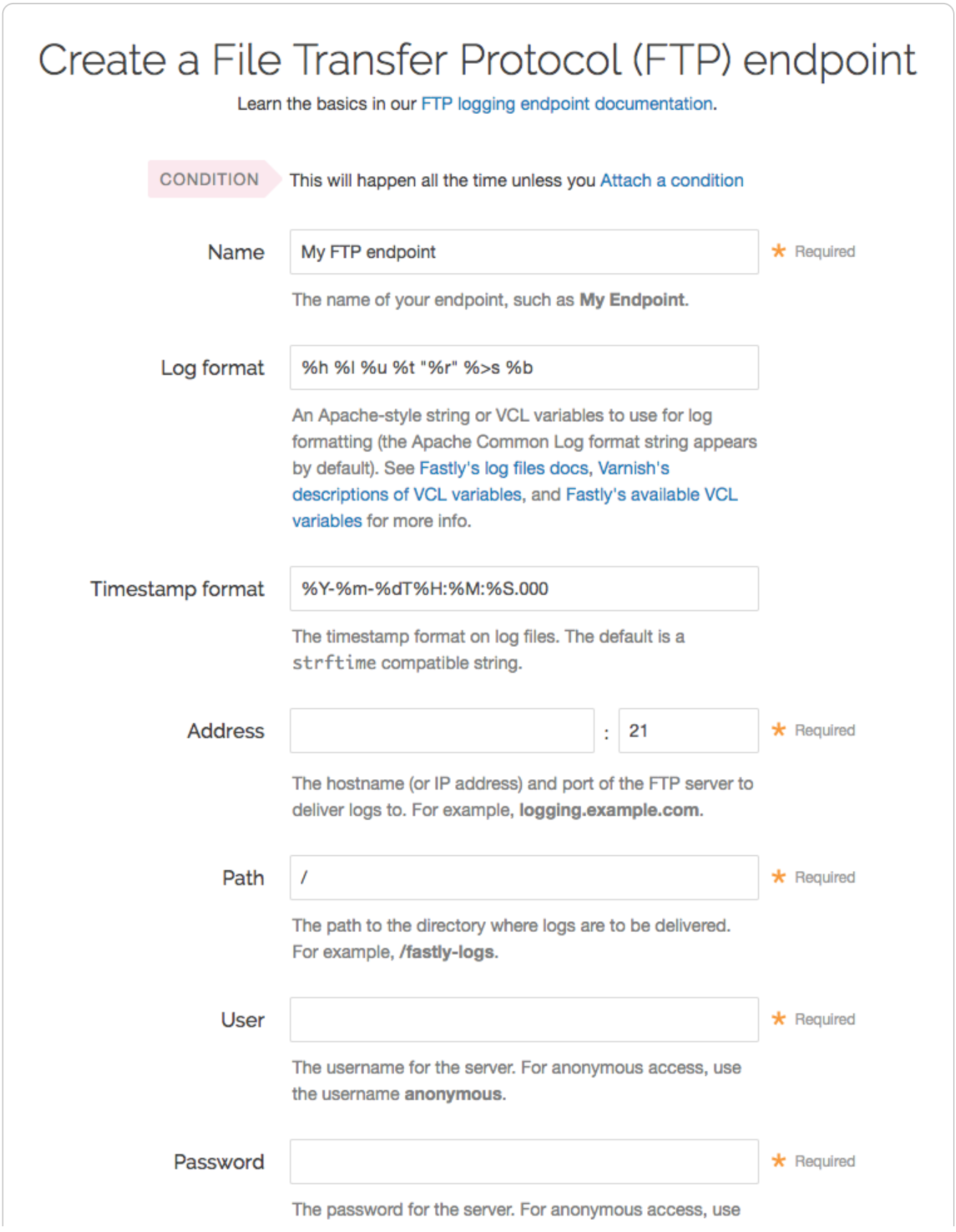
Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to password-protected and anonymous FTP servers.

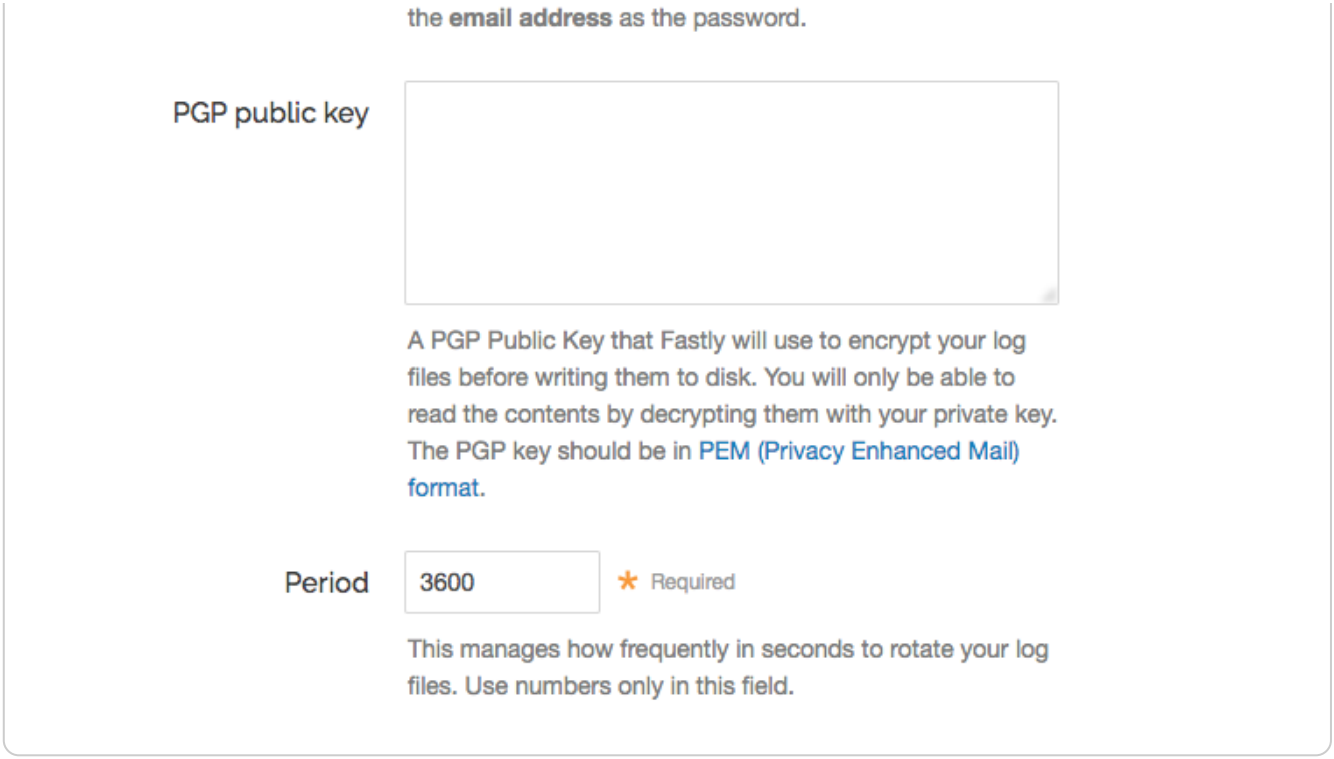
> ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Adding FTP as a logging endpoint

Follow these instructions to add FTP as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the FTP image. The Create a File Transfer Protocol (FTP) endpoint page appears.



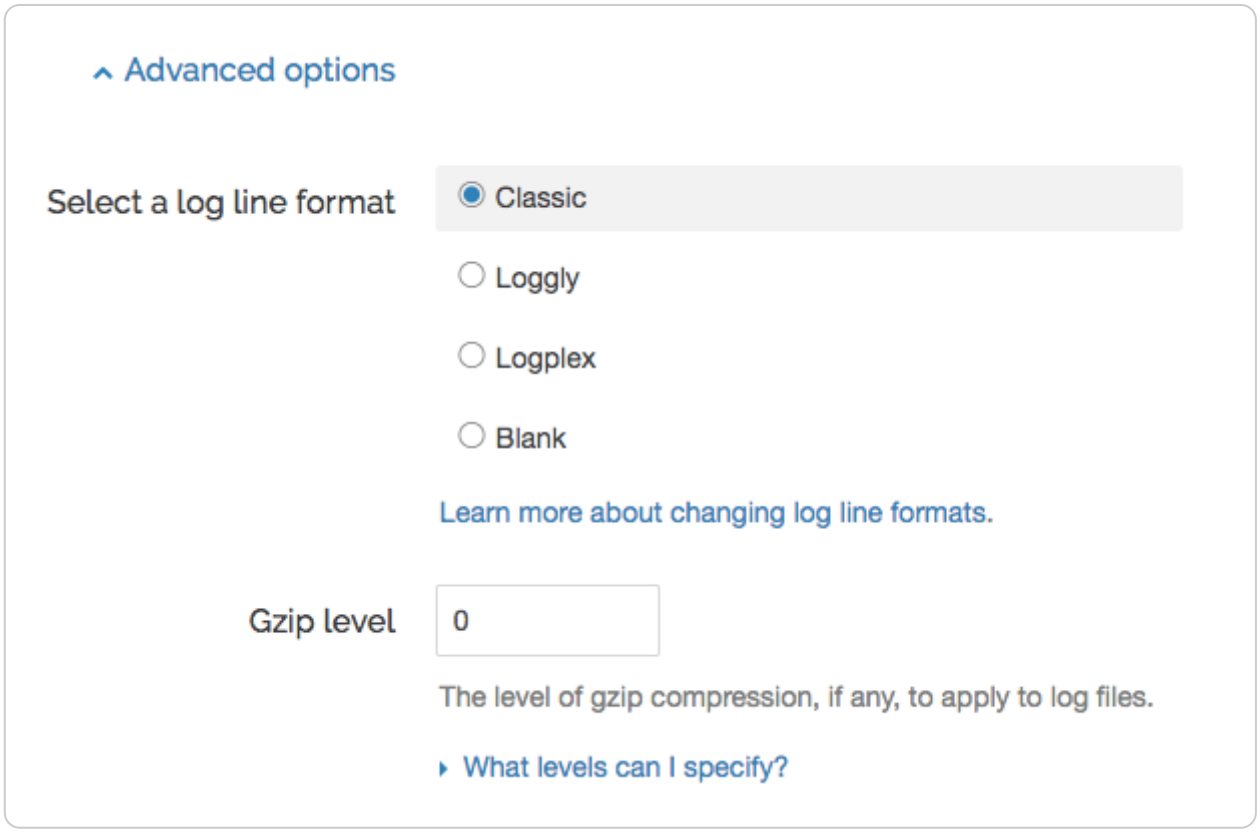


3. Fill out the **Create a File Transfer Protocol (FTP) endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
   - In the **Timestamp format** field, optionally type a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **Address** field, type the hostname or IP address of the FTP server. In the port field, type the port number you're using for FTP (the default is `21`).
   - In the **Path** field, optionally type the path to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the root path. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **User** field, type the username used to authenticate to the FTP server. For anonymous access, use the username `anonymous`.
   - In the **Password** field, type the password used to authenticate to the FTP server. For anonymous access, use an email address as the password.
   - In the **PGP public key** field, optionally type a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM

(Privacy-Enhanced Mail) format (https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail). See our guide on log encryption (/guides/streaming-logs/encrypting-logs) for more information.

- In the **Period** field, type an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a File Transfer Protocol (FTP) endpoint** page and decide which of the optional fields to change, if any.



5. Fill out the **Advanced options** of the **Create a File Transfer Protocol (FTP) endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats) provides more information.
   - In the **Gzip Level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

§ Log streaming: Google BigQuery (/guides/streaming-logs/log-streaming-google-bigquery)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to BigQuery (https://cloud.google.com/bigquery/), Google's managed enterprise data warehouse.

IMPORTANT: This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions. Fastly does not currently recommend streaming more than 20,000 log lines per second to this endpoint. Contact support@fastly.com (mailto:support@fastly.com) if your log needs will exceed this rate.

NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Prerequisites

Before adding BigQuery as a logging endpoint for Fastly services, you will need to:

- Register for a Google Cloud Platform (https://cloud.google.com/) (GCP) account.
- Create a service account on Google's website.
- Obtain the `private_key` and `client_email` from the JSON file associated with the service account.
- Enable the BigQuery API.
- Create a BigQuery dataset.
- Add a BigQuery table.

Creating a service account

BigQuery uses service accounts for third-party application authentication. To create a new service account, see Google's guide on generating service account credentials (https://cloud.google.com/storage/docs/authentication#generating-a-private-key). When you create the service account, set the key type to JSON.

Obtaining the private key and client email

After you create the service account, download the JSON file to your computer. This file contains the credentials for your BigQuery service account. Open the file and make a note of the `private_key` and `client_email`.

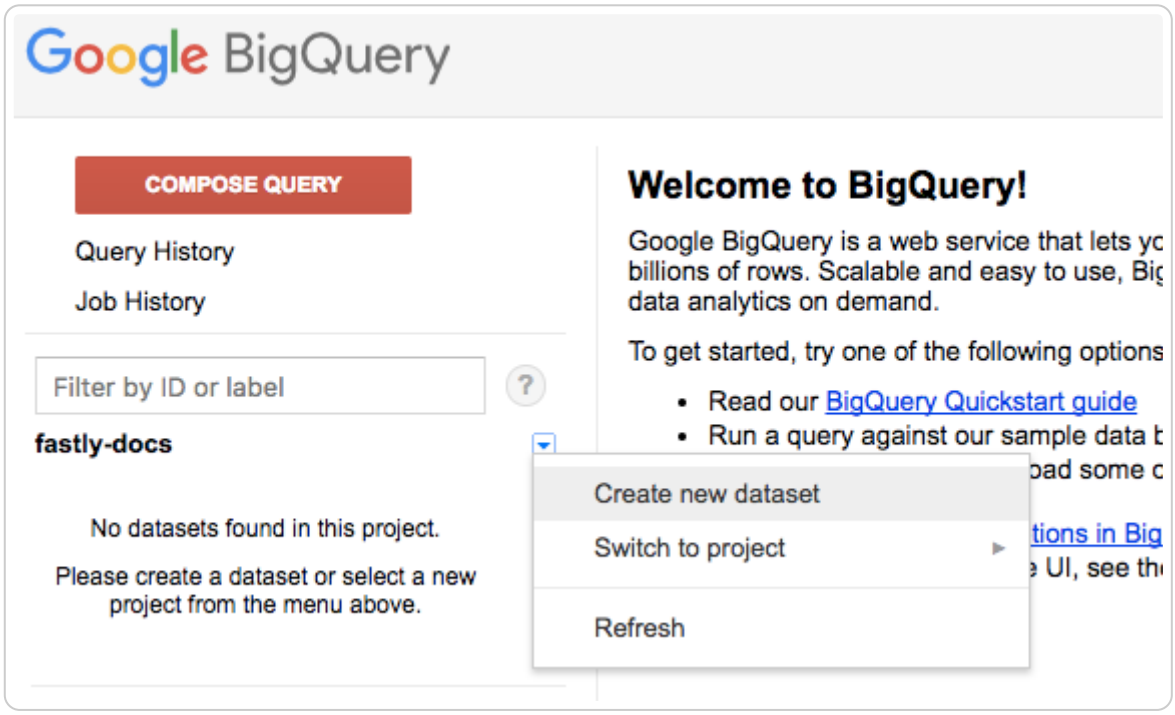
Enabling the BigQuery API

To send your Fastly logs to your BigQuery table, you'll need to enable the BigQuery API in the Google Cloud Platform API Manager (https://console.cloud.google.com/apis/library).

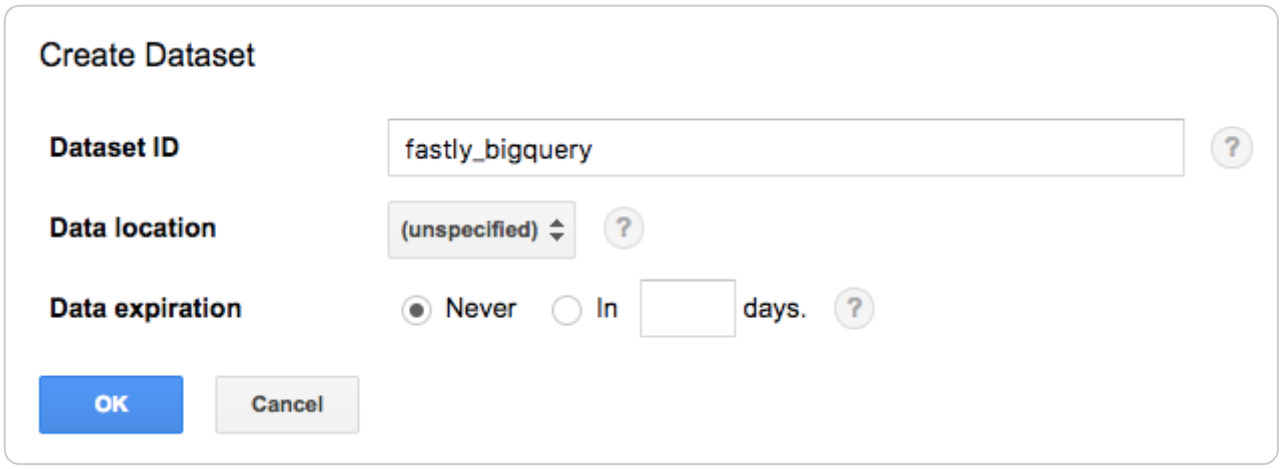
Creating the BigQuery dataset

After you've enabled the BigQuery API, follow these instructions to create a BigQuery dataset:

1. Log in to BigQuery (https://cloud.google.com/bigquery/).
2. Click the arrow next to your account name on the sidebar and select **Create new dataset**.



The Create Dataset window appears.



3. In the **Dataset ID** field, enter a name for the dataset (e.g., `fastly_bigquery`).
4. Click the **OK** button.

Adding a BigQuery table

After you've created the BigQuery dataset, you'll need to add a BigQuery table. There are four ways of creating the schema for the table:

- Edit the schema using the BigQuery web interface.
- Edit the schema using the text field in the BigQuery web interface.
- Use an existing table.
- Set the table to automatically detect the schema (https://cloud.google.com/bigquery/docs/schema-detect).

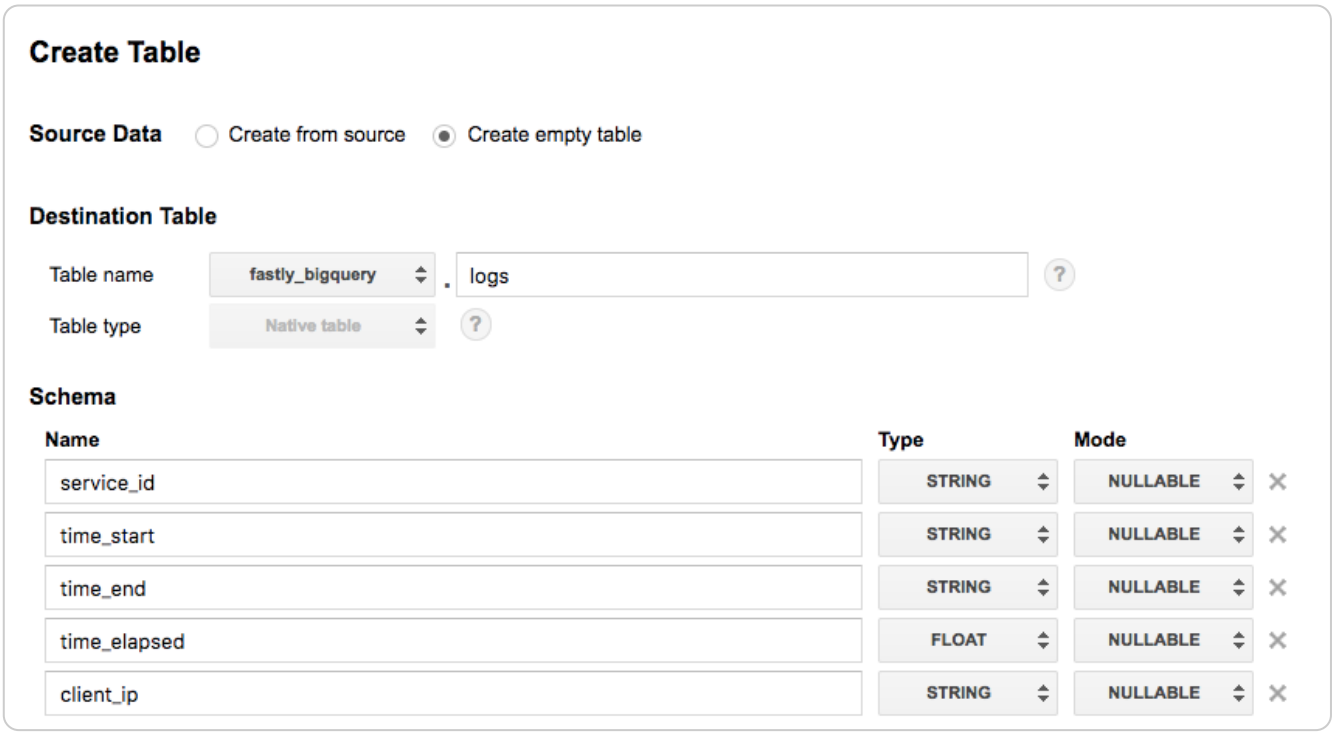
> ℹ **NOTE:** Setting the table to automatically detect the schema may give unpredictable results.

Follow these instructions to add a BigQuery table:

1. On the BigQuery website (https://cloud.google.com/bigquery/), click the arrow next to the dataset name on the sidebar and select **Create new table**.



The Create Table page appears.

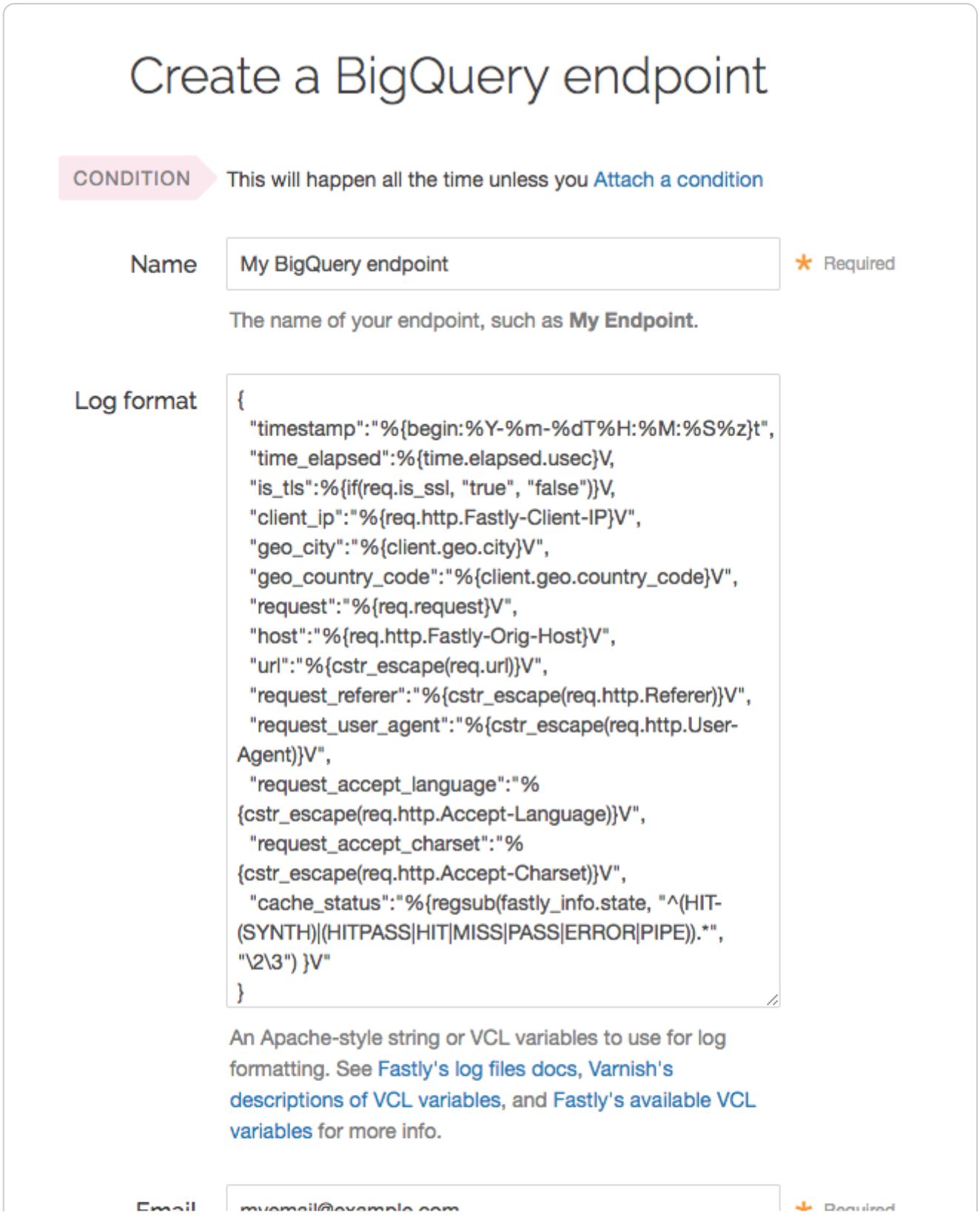


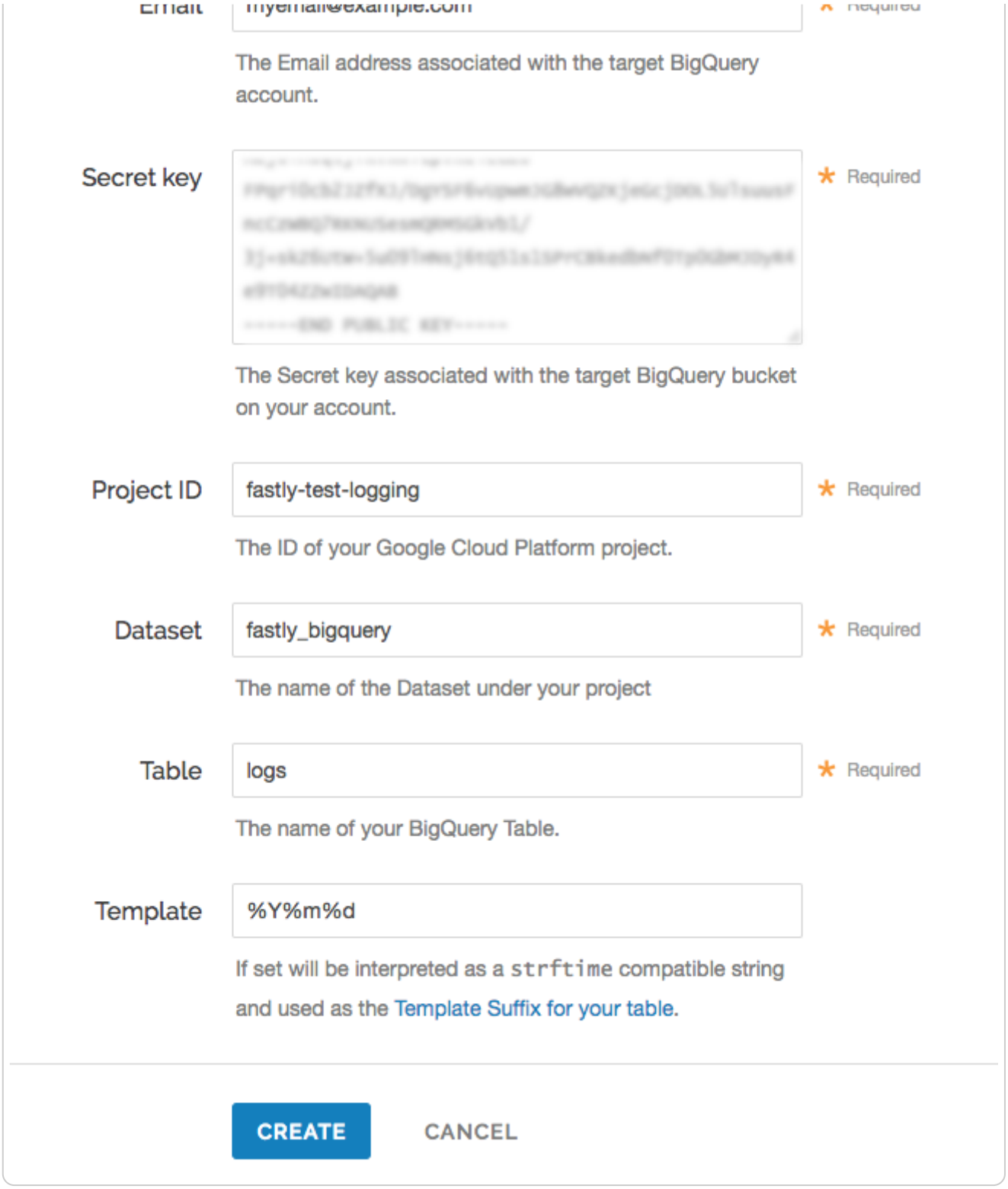
2. In the **Source Data** section, select **Create empty table**.
3. In the **Table name** field, enter a name for the table (e.g., `logs`).
4. In the **Schema** section of the BigQuery website, use the interface to add fields and complete the schema. See the example schema section for details.
5. Create the **Create Table** button.

Adding BigQuery as a logging endpoint

Follow these instructions to add BigQuery as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the BigQuery logo. The Create a BigQuery endpoint page appears.





3. Fill out the **Create a BigQuery endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Log format** field, enter the data to send to BigQuery. See the example format section for details.
   - In the **Email** field, enter the `client_email` address associated with the BigQuery service account.

- In the **Secret key** field, enter the value of the `private_key` associated with your BigQuery service account.
- In the **Project ID** field, enter the ID of your Google Cloud Platform project.
- In the **Dataset** field, enter the name of your BigQuery dataset.
- In the **Table** field, enter the name of your BigQuery table.
- In the **Template** field, optionally enter an `strftime` compatible string to use as the template suffix for your table (https://cloud.google.com/bigquery/streaming-data-into-bigquery#template-tables).

4. Click **Create** to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

Data sent to BigQuery must be serialized as a JSON object, and every field in the JSON object must map to a string in your table's schema. The JSON can have nested data in it (e.g. the value of a key in your object can be another object). Here's an example format string for sending data to BigQuery:

```
{
  "timestamp":"%{begin:%Y-%m-%dT%H:%M:%S%z}t",
  "time_elapsed":%{time.elapsed.usec}V,
  "is_tls":%{if(req.is_ssl, "true", "false")}V,
  "client_ip":"%{req.http.Fastly-Client-IP}V",
  "geo_city":"%{client.geo.city}V",
  "geo_country_code":"%{client.geo.country_code}V",
  "request":"%{req.request}V",
  "host":"%{req.http.Fastly-Orig-Host}V",
  "url":"%{cstr_escape(req.url)}V",
  "request_referer":"%{cstr_escape(req.http.Referer)}V",
  "request_user_agent":"%{cstr_escape(req.http.User-Agent)}V",
  "request_accept_language":"%{cstr_escape(req.http.Accept-Language)}V",
  "request_accept_charset":"%{cstr_escape(req.http.Accept-Charset)}V",
  "cache_status":"%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIP
E)).*", "\\2\\3") }V"
}
```

Example schema

The BigQuery schema for the example format shown above would look something like this:

```
timestamp:STRING,time_elapsed:FLOAT,is_tls:BOOLEAN,client_ip:STRING,geo_city:STRING,geo_country
_code:STRING,request:STRING,host:STRING,url:STRING,request_referer:STRING,request_user_agent:ST
RING,request_accept_language:STRING,request_accept_charset:STRING,cache_status:STRING
```

§ Log streaming: Google Cloud Storage (/guides/streaming-logs/log-streaming-google-cloud-storage)

Fastly's Real-Time Log Streaming feature (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) can send log files to Google Cloud Storage (https://cloud.google.com/storage/) (GCS). GCS is an online file storage service used for storing and accessing data on Google's infrastructure. One advantage of using GCS is that you can use Google BigQuery (https://cloud.google.com/bigquery/) to analyze the log files.

> ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Prerequisites

Before adding GCS as a logging endpoint for Fastly services, you will need to:

- Register for a GCS account.
- Create a bucket and service account on Google's website.
- Obtain the `private_key` and `client_email` from the JSON file associated with the service account.
- Enable the Google Cloud Storage JSON API.

Creating a GCS bucket

You can create a new GCS bucket to hold the logs, or you can use an existing bucket. Be sure to note the name of the bucket as you will need it later. To learn how to create a GCS bucket, see Google's guide on creating a bucket (https://cloud.google.com/storage/docs/getting-started-console#create_a_bucket).

Creating a service account

GCS uses service accounts for third-party application authentication. You will need to create a new service account on Google's website. To learn how to create a service account, see Google's guide on generating a service account credential (https://cloud.google.com/storage/docs/authentication#generating-a-private-key). When you create the service account, be sure to set the **Key Type** to `JSON`.

Obtaining the private key and client email

After you create the service account, a JSON file will be downloaded to your computer. This file contains the credentials for the GCS service account you just created. Open the file with a text editor and make a note of the `private_key` and `client_email`.

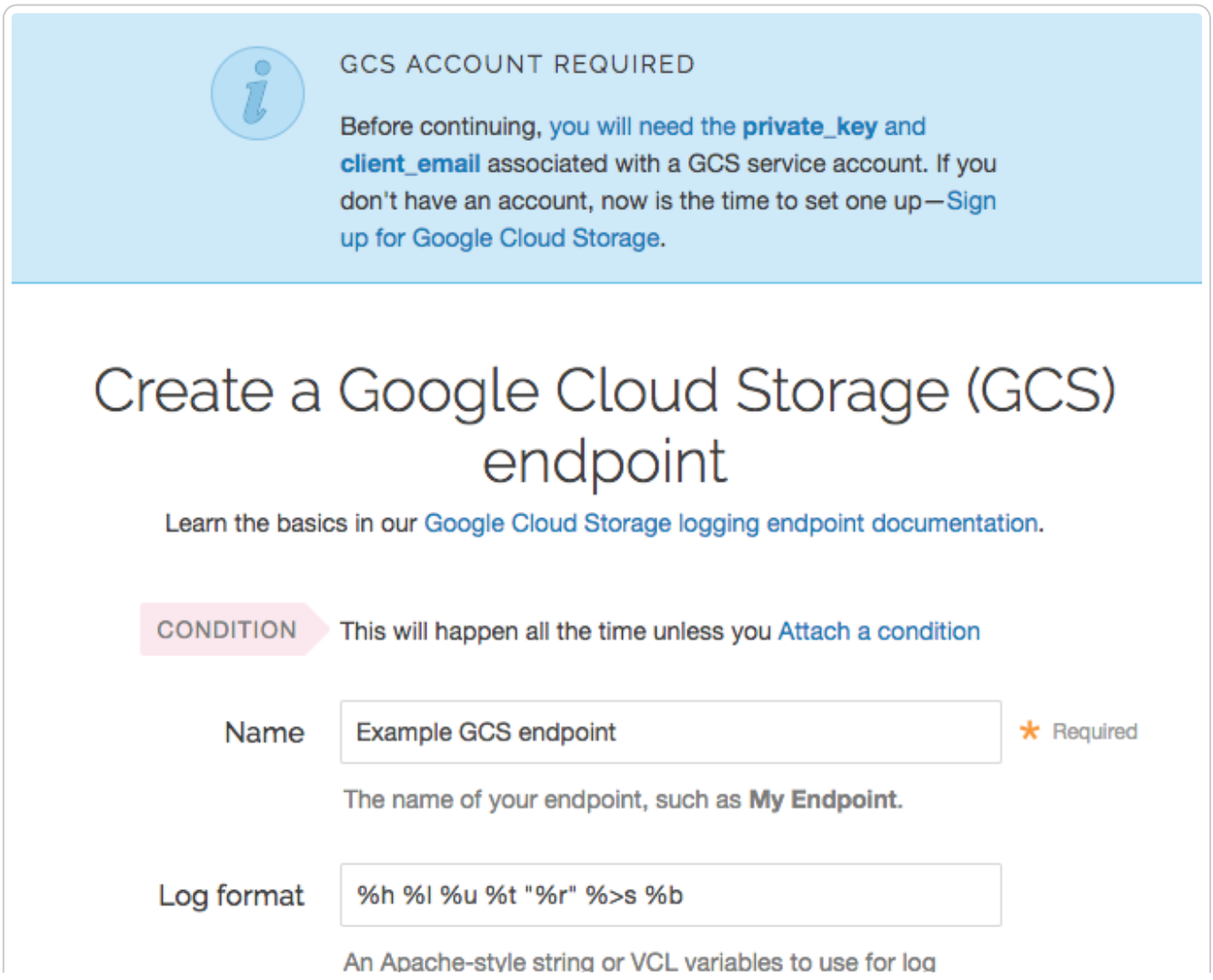
Enabling the Google Cloud Storage JSON API

To ensure the Fastly logs are sent to your GCS bucket, you need to enable the Google Cloud Storage JSON API. For more information, see Google's instructions for activating the API (https://cloud.google.com/storage/docs/json_api/).

Adding GCS as a logging endpoint

Follow these instructions to add GCS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Google Cloud Services logo. The Create a Google Cloud Storage (GCS) endpoint page appears.



formatting (the Apache Common Log format string appears by default). See Fastly's log files docs, Varnish's descriptions of VCL variables, and Fastly's available VCL variables for more info.

Timestamp format

%Y-%m-%dT%H:%M:%S.000

The timestamp format on log files. The default is a `strftime` compatible string.

Email

* Required

The Email address associated with the target GCS bucket on your account.

Bucket name

* Required

The name of the bucket in which to store the logs.

Secret key

* Required

The Secret key associated with the target GCS bucket on your account.

PGP public key

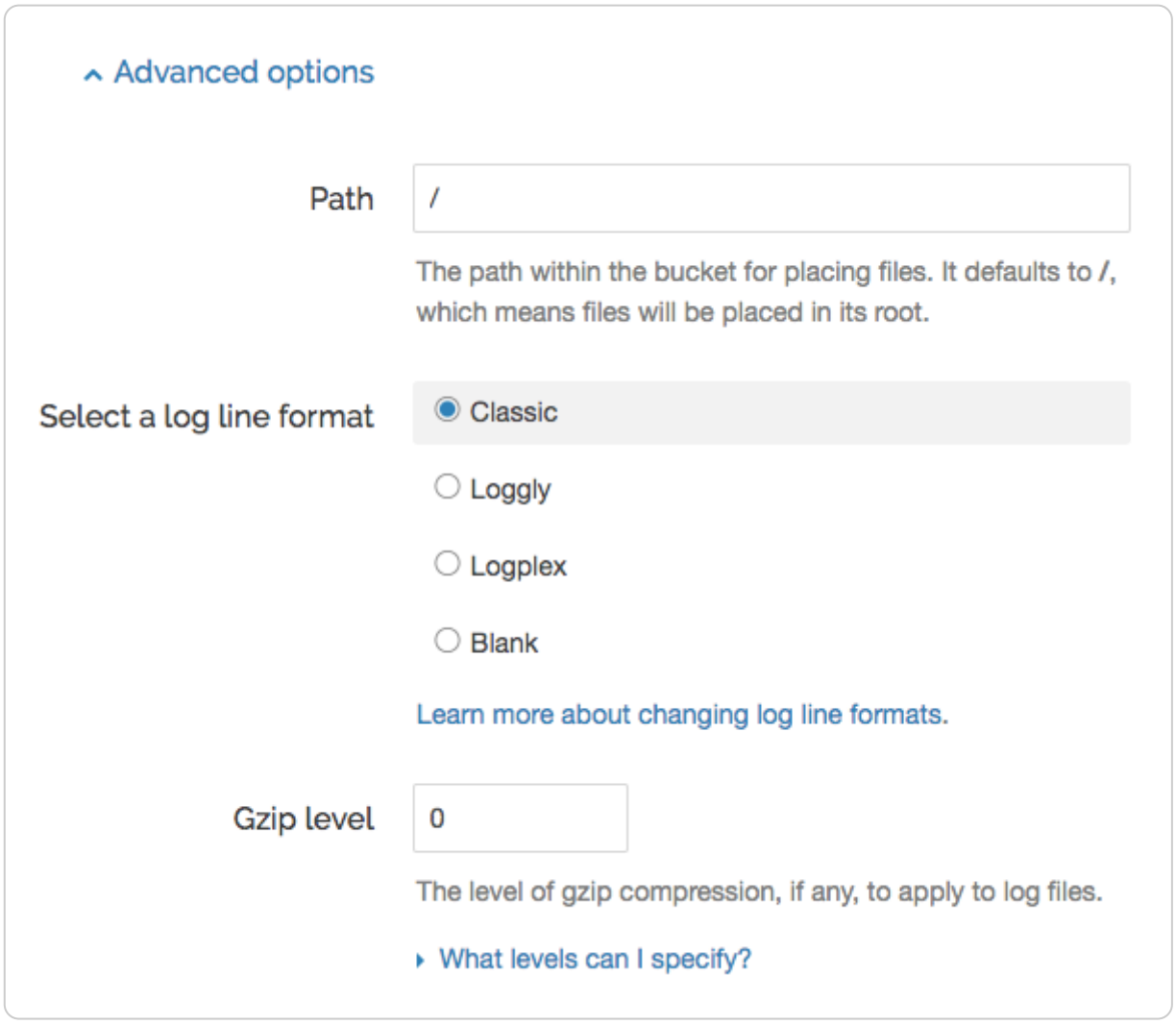
A PGP Public Key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy Enhanced Mail) format.

Period

3600 * Required

This manages how frequently in seconds to rotate your log files. Use numbers only in this field.

3. Fill out the **Create a Google Cloud Storage (GCS) endpoint** fields as follows:
    - In the **Name** field, type a human-readable name for the endpoint.
    - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings (/guides/streaming-logs/custom-log-formats) for more information.
    - In the **Timestamp format** field, optionally type a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
    - In the **Email** field, type the `client_email` address listed in the JSON file associated with the service account you created on Google's website.
    - In the **Bucket name** field, type the name of the GCS bucket in which to store the logs.
    - In the **Secret key** field, type the `private_key` value listed in the JSON file associated with the service account you created on Google's website. We strip out the JSON newline escape characters for you so don't worry about removing them.
    - In the **PGP public key** field, optionally type a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format (https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail). See our guide on log encryption (/guides/streaming-logs/encrypting-logs) for more information.
    - In the **Period** field, optionally type an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a Google Cloud Storage (GCS) endpoint** page and decide which of the optional fields to change, if any.



5. Fill out the **Advanced options** of the **Create a Google Cloud Storage (GCS) endpoint** page as follows:
   - In the **Path** field, optionally type the path within the bucket to store the files. Specify a directory by ending the path with a trailing slash (`/`). Leaving this field empty saves the files in the bucket's root path. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats) provides more information.
   - In the **Gzip Level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
6. Click **Create** to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

§ Log streaming: Log Shuttle (/guides/streaming-logs/log-streaming-log-shuttle)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to Log Shuttle (https://github.com/heroku/log-shuttle). Log Shuttle is an open source application designed to provide simpler encrypted and authenticated log delivery.

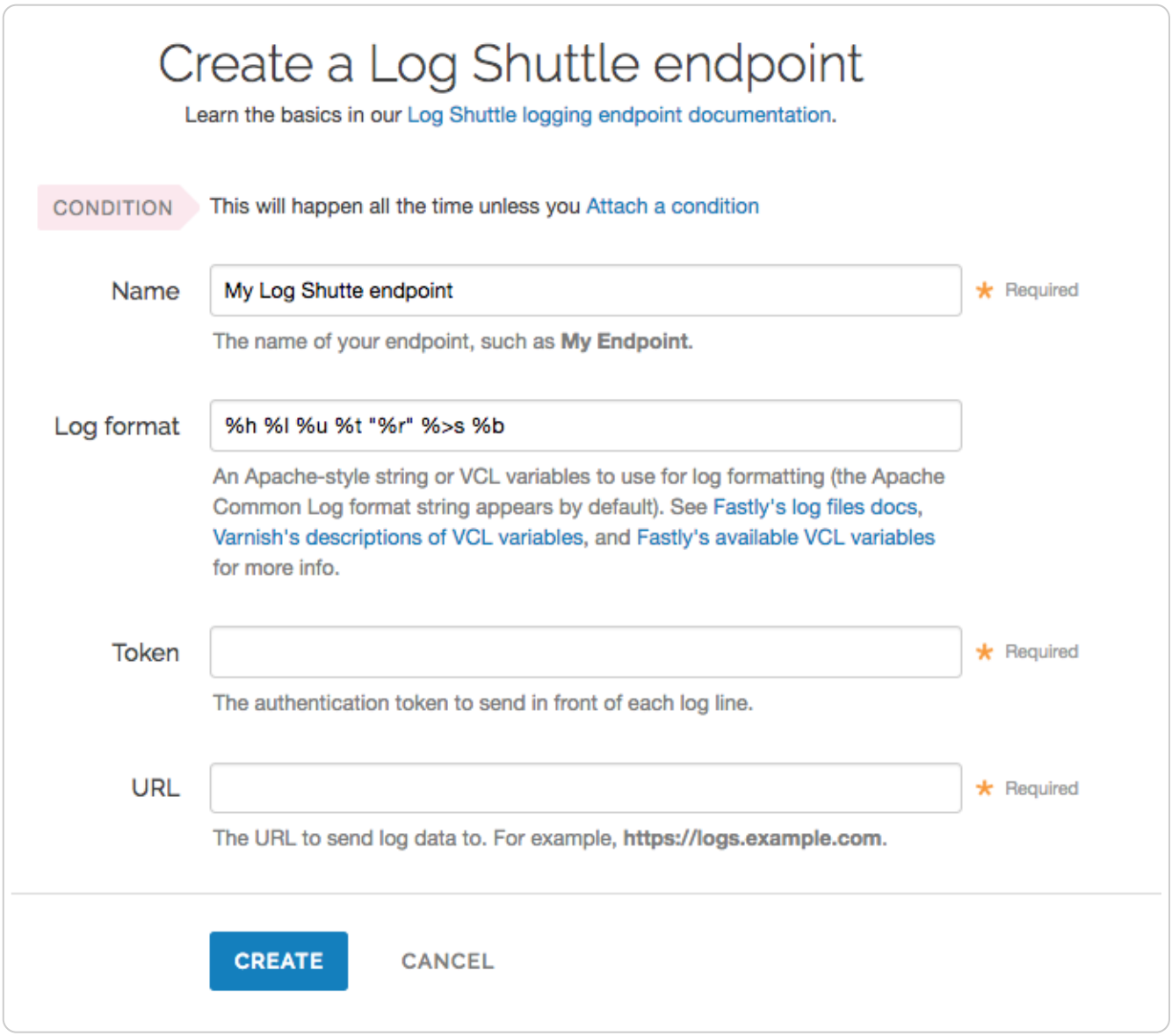
ℹ **NOTE:** This logging endpoint is disabled by default. To enable this endpoint for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Adding Log Shuttle as a logging endpoint

After Fastly support has enabled the Log Shuttle endpoint for your account, follow these instructions to add Log Shuttle as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Log Shuttle logo. The Create a Log Shuttle endpoint page appears.



3. Fill out the **Create a Log Shuttle endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
   - In the **Token** field, type the data authentication token. This is required for some endpoints like Heroku's Log Integration.
   - In the **URL** field, type the URL to which log data will be sent (e.g., `https://logs.example.com/`).
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

§ Log streaming: Logentries (/guides/streaming-logs/log-streaming-logentries)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to Logentries (https://logentries.com/). Logentries is a real-time log management and analytics system that you can use to monitor your Fastly logs.

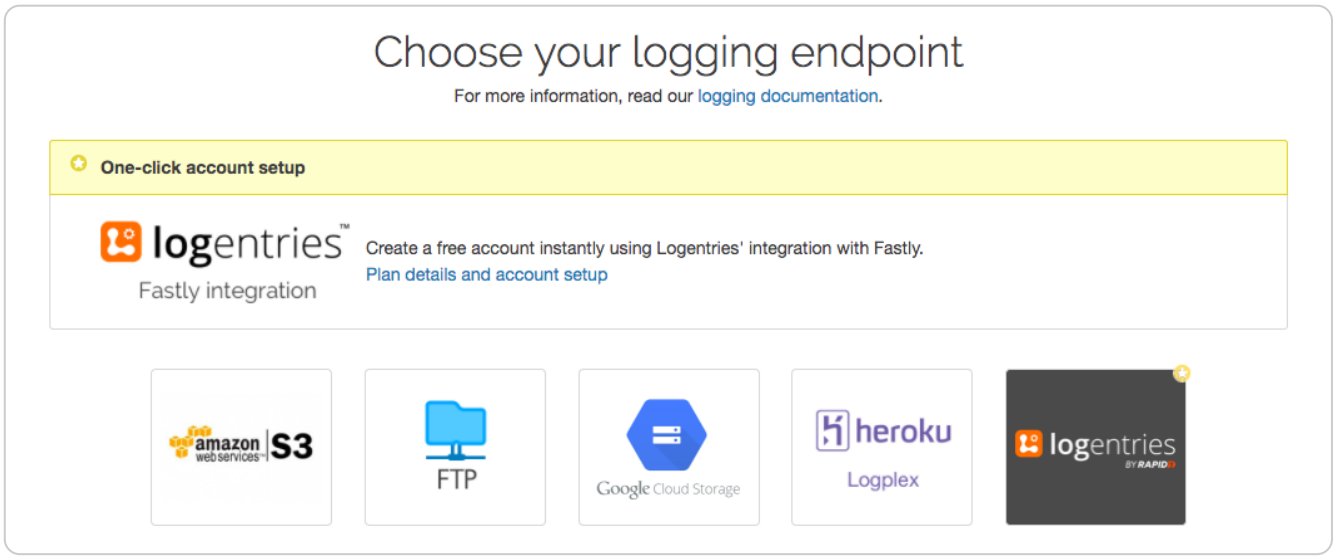
> ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

One-click Logentries account setup

Fastly has partnered with Logentries to offer you a method for automatically creating a Logentries account and configuring a logging endpoint. By using the Logentries one-click integration, you can create a 30 day trial Logentries account with unlimited data. After 30 days, if you don't upgrade to one of the Logentries premium plans (https://logentries.com/), your account will be capped at 5GB per month.

Follow these instructions to create a Logentries logging endpoint and configure the logging endpoint:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears. If you have an existing logging endpoint, click the **Create endpoint** button.

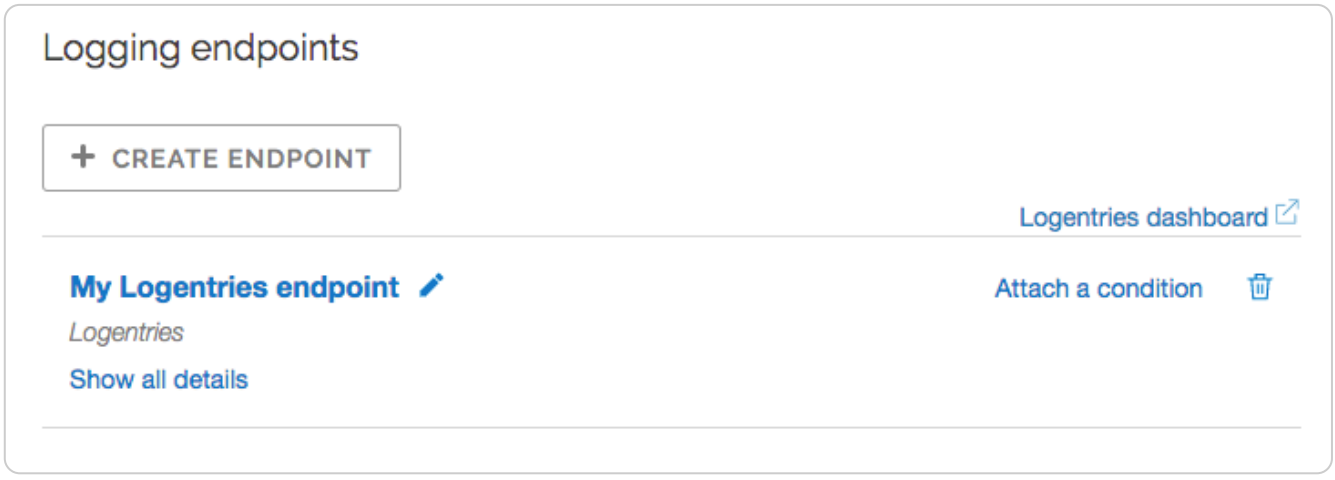


5. In the One-click account setup box, click the **Plan details and account setup** link. The Create Logentries Account window appears.
6. Click the **Create Logentries Account** button.
7. Click the **Activate** button to deploy your configuration changes.

Accessing your Logentries account

If you created a Logentries account using the one-click integration, you must access your Logentries account from the Fastly web application. Follow these instructions to log in to Logentries:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the Logging link. The Logging endpoints page appears.



5. Click the **Logentries dashboard** link to access your Logentries account dashboard.

Manually adding Logentries as a logging endpoint

If you already have a Logentries account, or if you'd prefer to sign up for a Logentries account on the Logentries website, you can manually add Logentries as a logging endpoint in the Fastly web interface.

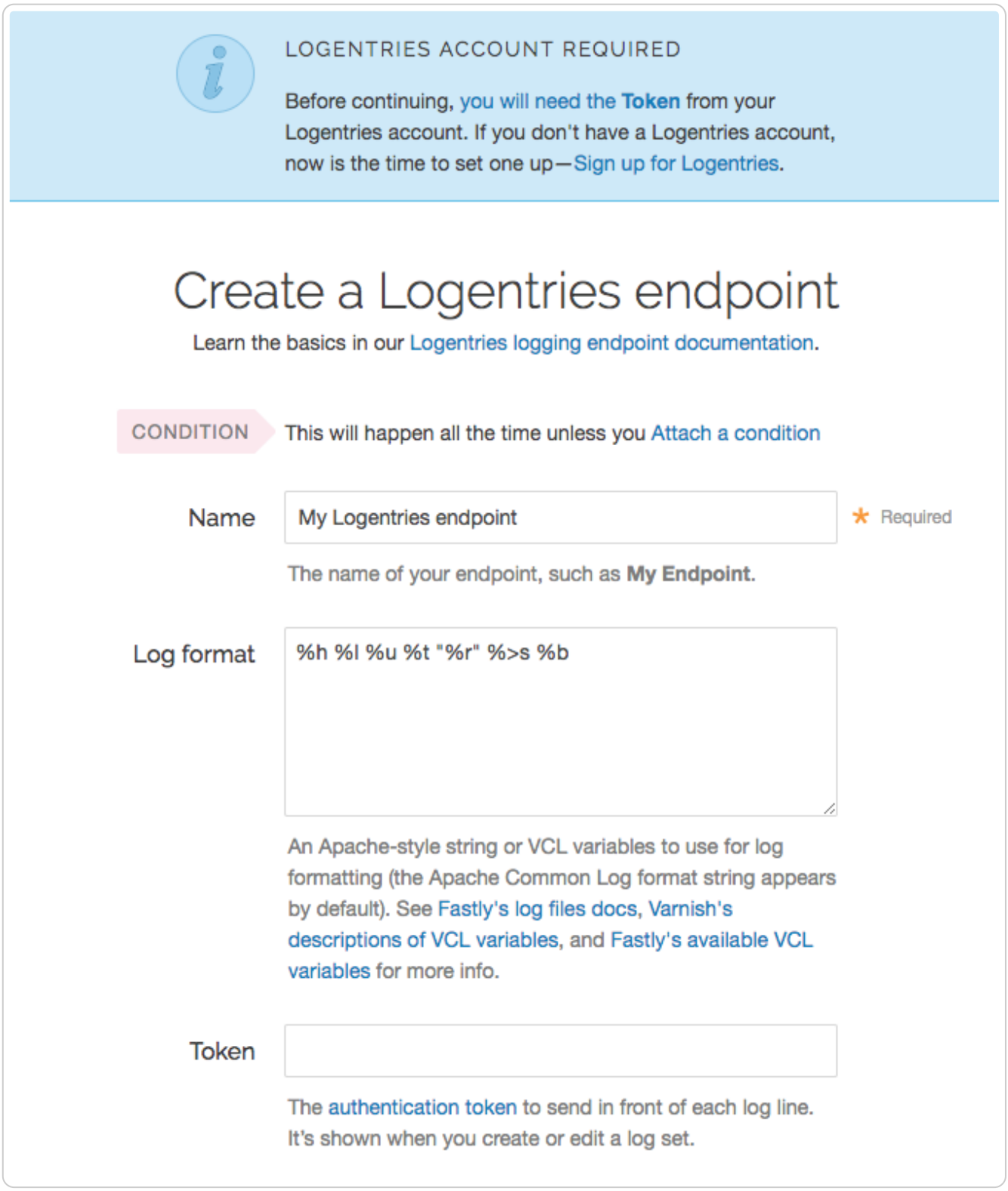
Prerequisites

1. Register for a Logentries (https://logentries.com/) account.
2. Create a new log in the Logentries application by following the instructions on the Logentries website (https://logentries.com/doc/fastly/)
3. During new log creation, select **Manual Configuration** and **Token TCP**.
4. Make a note of the token provided in the Logentries configuration panel. We recommend you use this token when you create the Logentries logging endpoint for Fastly services.

Creating the logging endpoint in the web interface

After you've created a new log in Logentries and found the token, follow these instructions to add Logentries as a logging endpoint for Fastly services:

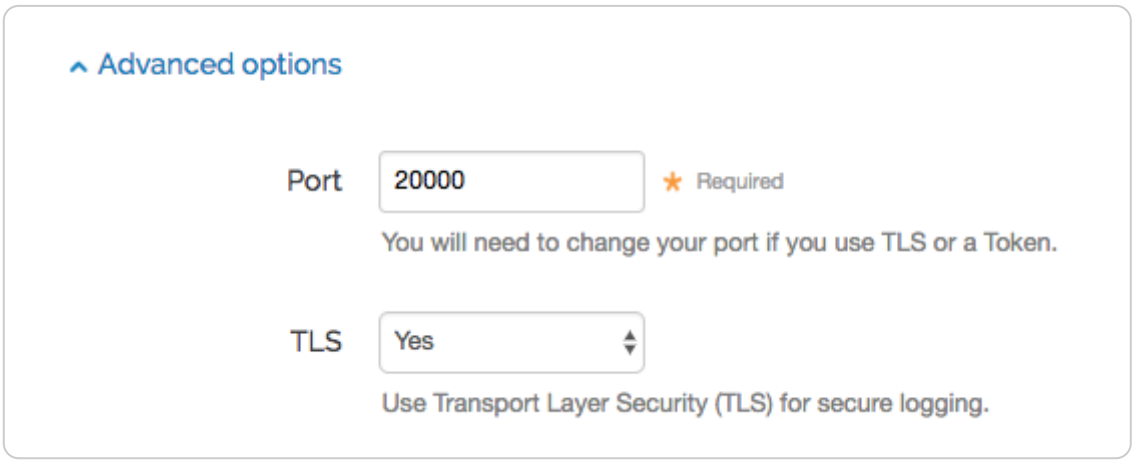
1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Logentries logo. The Create a Logentries endpoint page appears.



3. Fill out the **Create a Logentries endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. Our discussion of format strings (/guides/streaming-logs/custom-log-formats) also provides more information.

- In the **Token** field, type the token provided in the Logentries configuration panel. Though you can use the provided secret port number, there are additional options to consider when deciding on token settings.

4. Click the **Advanced options** link.



5. Fill out the **Advanced options** as follows:
   - In the **Port** field, type `20000`. Though we recommend this specific setting when adding your endpoint, there are additional options to consider when deciding on the port and TLS settings.
   - From the **TLS** menu, optionally select **Yes**.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Additional selections for tokens, ports, and TLS

Using your token. You can add a Logentries endpoint by using your secret account token. To use your token, set your port to `10000`. However, we strongly recommend sending your logs via TLS. To do this, set TLS to `Yes` and the port number to `20000`. See the Logentries guide Token TCP (https://docs.logentries.com/docs/input-token) for more information.

Using your port number. You can add a Logentries endpoint by using your secret Logentries port number (e.g., `56789`). However, we strongly recommend sending your logs via TLS. To do this, set TLS to `Yes` and add `10000` to your secret port number (e.g., `66789`). See the Logentries guide Plain TCP/UDP (https://docs.logentries.com/docs/input-plaintcpudp) for more information.

Next steps

Logentries maintains the Fastly Community Pack (https://blog.logentries.com/2015/01/fastly-community-pack/) that leverages custom VCL to provide advanced User-Agent statistics, regional statistics, error tracking, and more.

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

§ Log streaming: Loggly (/guides/streaming-logs/log-streaming-loggly)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to Loggly (https://www.loggly.com/). Loggly is an agent-less log collection and management tool.

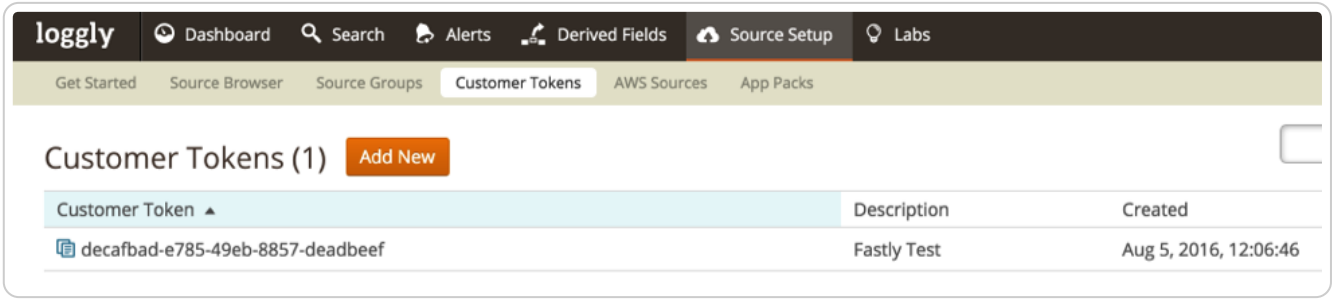
NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Prerequisites

If you don't already have a Loggly account, you'll need to register for one. Follow the signup instructions (https://www.loggly.com/signup/) on the Loggly website.

Follow the steps below to find your Loggly customer token:

1. Navigate to the **Customer Tokens (https://www.loggly.com/docs/customer-token-authentication-token/)** area in the **Source Setup** on your Loggly dashboard.

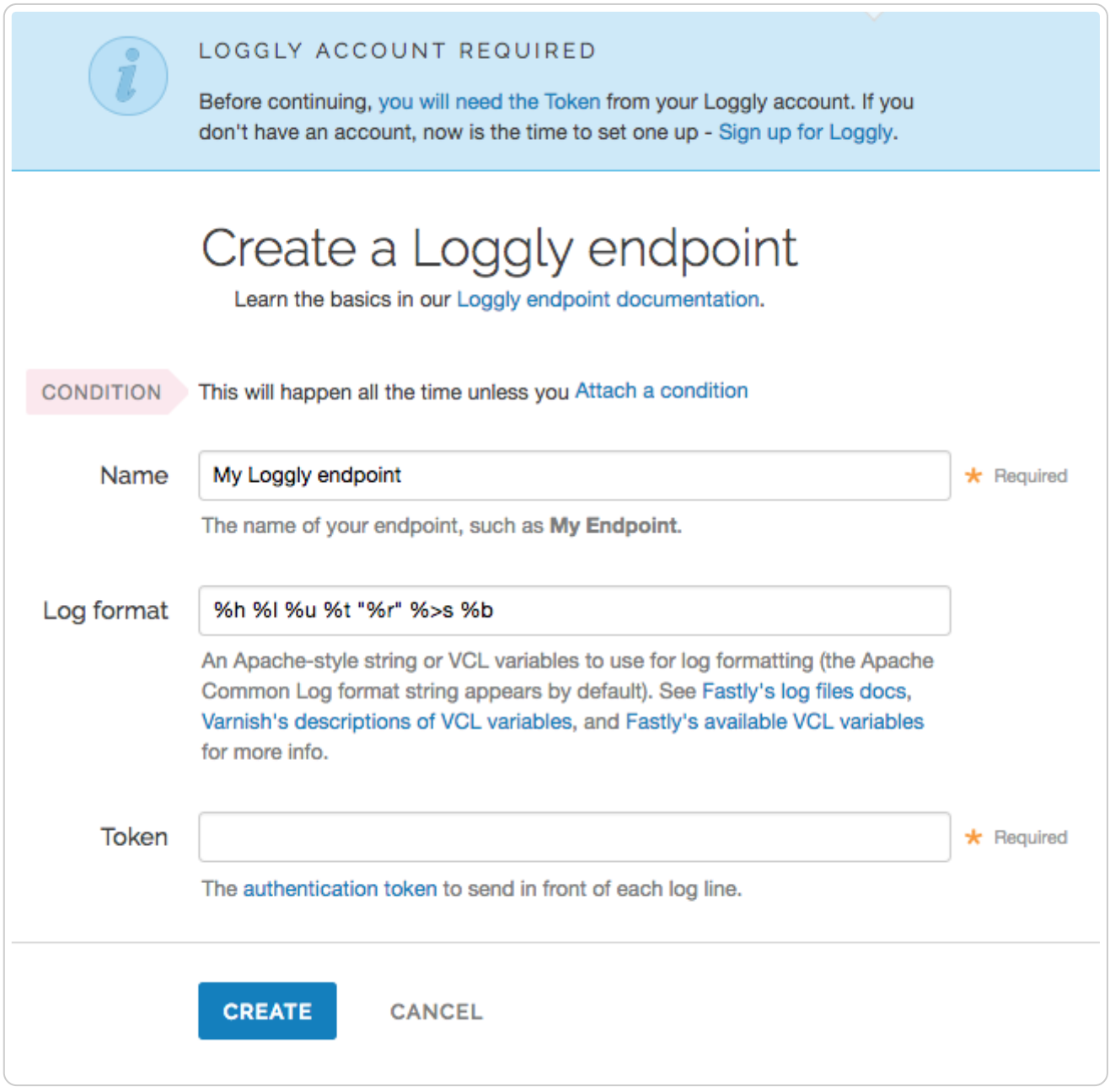


2. Make note of your Loggly customer token. Loggly uses this to associate data you send them with your account.

Adding Loggly as a logging endpoint

After you've created a Loggly account and obtained your customer token, follow these instructions to add Loggly as a logging endpoint for Fastly services:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Loggly logo. The Create a Loggly endpoint page appears.



3. Fill out the **Create a Loggly endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. Our discussion of format strings (/guides/streaming-logs/custom-log-formats) provides more information.

- In the **Token** field, type your Loggly customer token.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

§ Log streaming: Heroku's Logplex (/guides/streaming-logs/log-streaming-logplex)

As part of our Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature, if you use our Heroku add-on (https://elements.heroku.com/addons/fastly), you can send log files directly to Heroku's Logplex (https://devcenter.heroku.com/articles/logplex) system. Logplex is Heroku's distributed syslog router that collates and distributes log entries from a variety of sources into a single channel.

❗ **IMPORTANT:** This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

To enable this feature for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Once enabled, your Fastly logs will be available in exactly the same way (https://devcenter.heroku.com/articles/logging) as your regular app and hosted service logs. You can view them using the Heroku command line log viewer or send them to a logging add-on (https://elements.heroku.com/addons/#logging).

§ Log streaming: OpenStack (/guides/streaming-logs/log-streaming-openstack)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to OpenStack (http://www.openstack.org/). OpenStack is an open-source platform for cloud-computing that many companies deploy as an infrastructure-as-a-service.

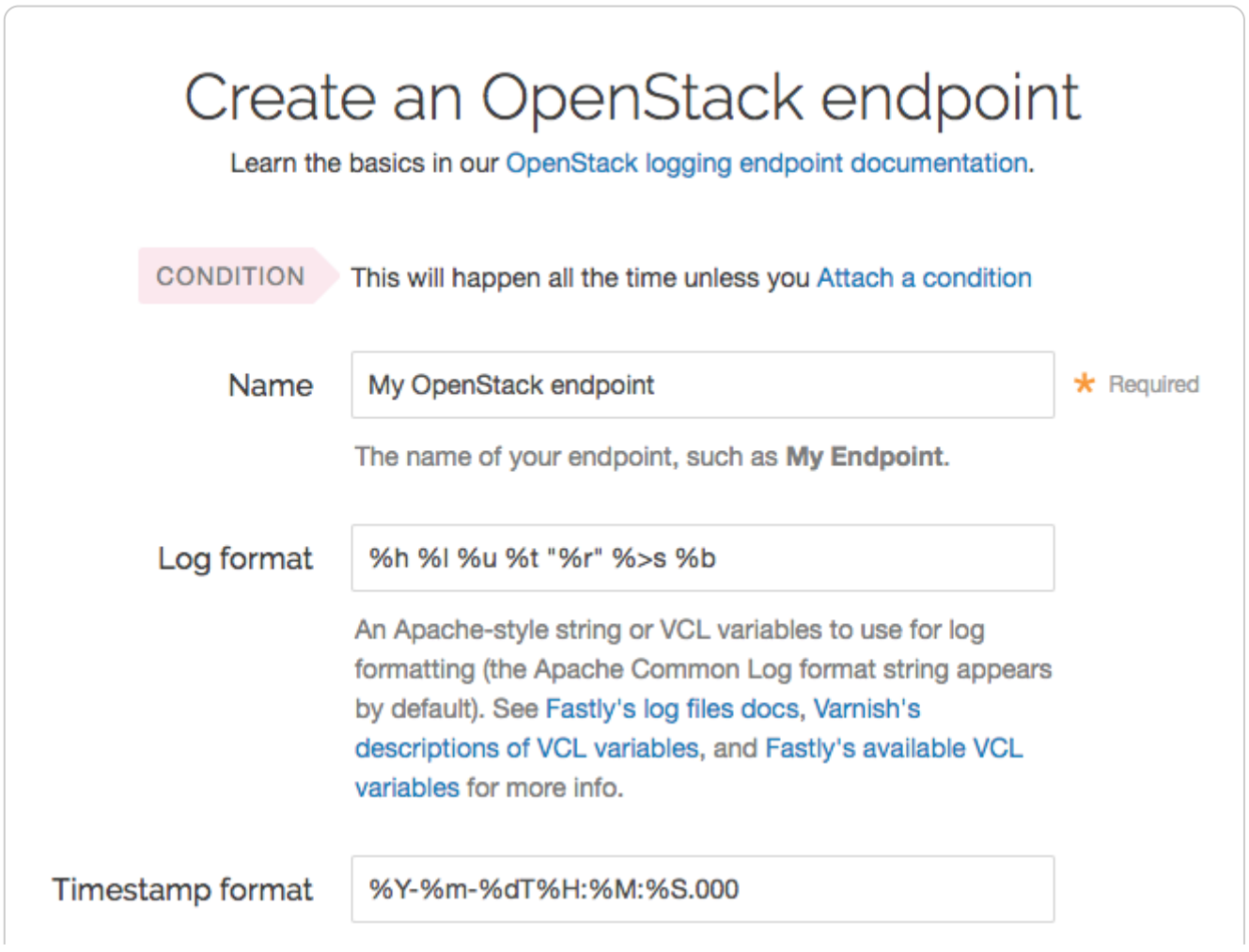
NOTE: This logging endpoint is disabled by default. To enable this endpoint for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

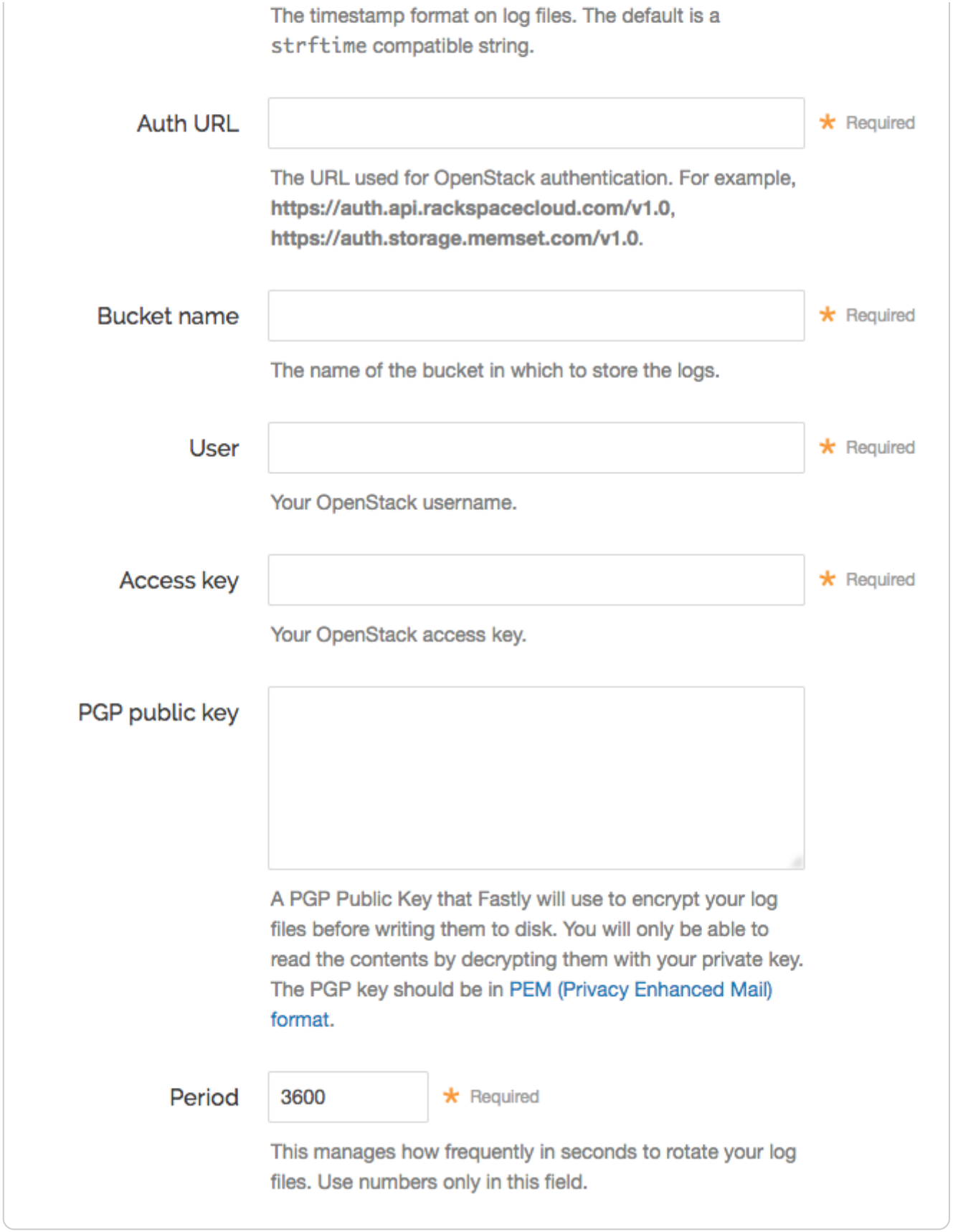
NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Adding OpenStack as a logging endpoint

After Fastly support has enabled the OpenStack endpoint for your account, follow these instructions to add OpenStack as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the OpenStack logo. The Create an OpenStack endpoint page appears.



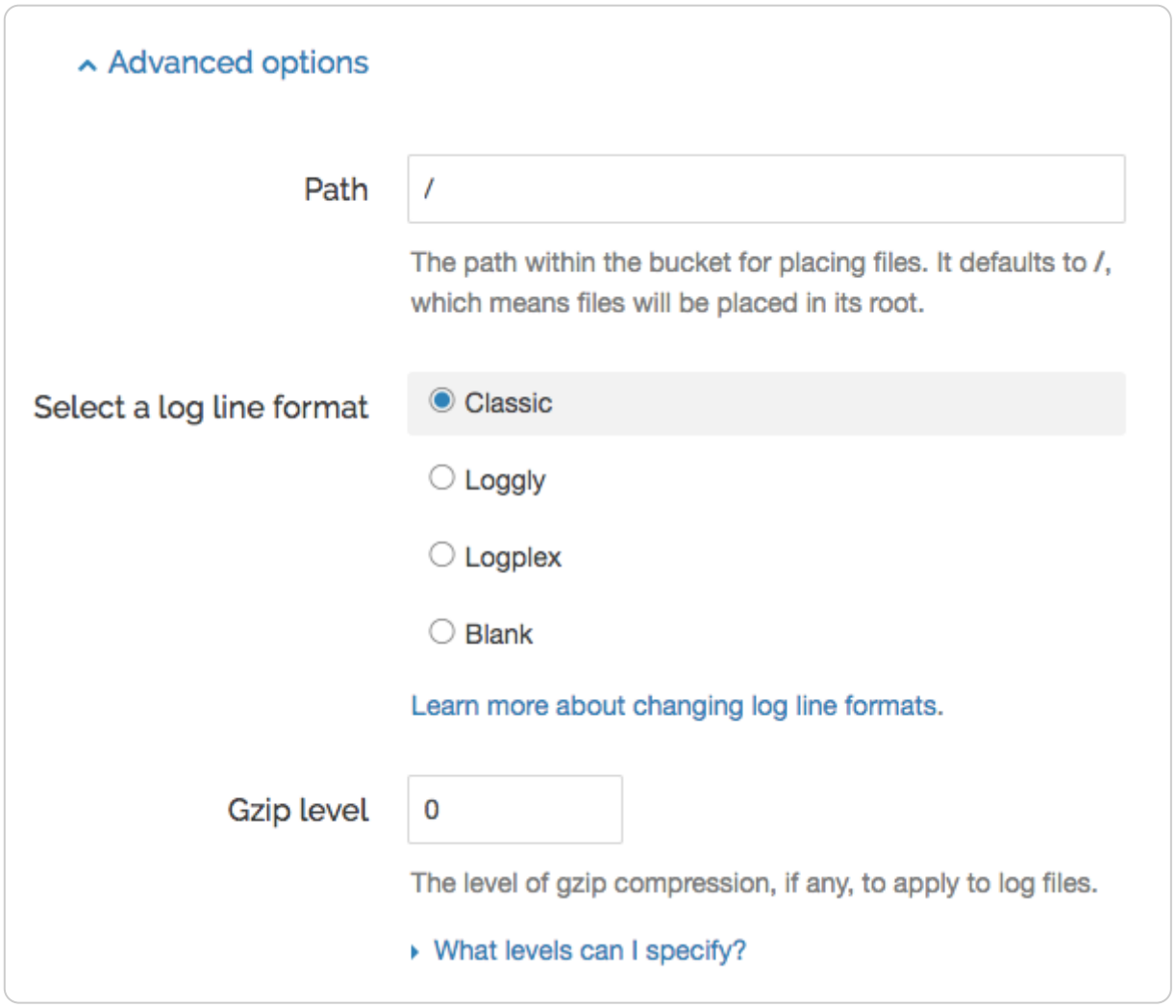


3. Fill out the **Create an OpenStack endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.

See our guidance on format strings (/guides/streaming-logs/custom-log-formats) for more information.

- In the **Timestamp format** field, optionally type a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
- In the **Auth URL** field, type the URL used for OpenStack authentication (e.g., `https://auth.api.rackspacecloud.com/v1.0`).
- In the **Bucket name** field, type the name of the OpenStack bucket in which to store the logs.
- In the **Username** field, type your OpenStack username.
- In the **Access Key** field, type your OpenStack access key.
- In the **PGP public key** field, optionally type a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format (https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail). See our guide on log encryption (/guides/streaming-logs/encrypting-logs) for more information.
- In the **Period** field, type an interval (in seconds) to control how frequently to rotate your log files. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a new OpenStack endpoint** page and decide which of the optional fields to change, if any.



5. Fill out the **Advanced options** of the **Create an OpenStack endpoint** page as follows:
   - In the **Path** field, optionally type the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats) provides more information.
   - In the **Gzip Level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

§ Log streaming: Papertrail (/guides/streaming-logs/log-streaming-papertrail)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to Papertrail (https://papertrailapp.com). Papertrail is a web-based log aggregation application used by developers and IT teams. Instructions for setting up remote log streaming via Papertrail are detailed in the Papertrail setup and configuration documentation (http://help.papertrailapp.com/kb/hosting-services/fastly/).

ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

§ Log streaming: Scalyr (/guides/streaming-logs/log-streaming-scalyr)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to Scalyr (https://www.scalyr.com/). Scalyr pulls all your server logs and metrics into a centralized, searchable system in real-time.

ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Prerequisites

If you don't already have a Scalyr account, you'll need to register for one. Follow the signup instructions (https://www.scalyr.com/signup) on the Scalyr website.

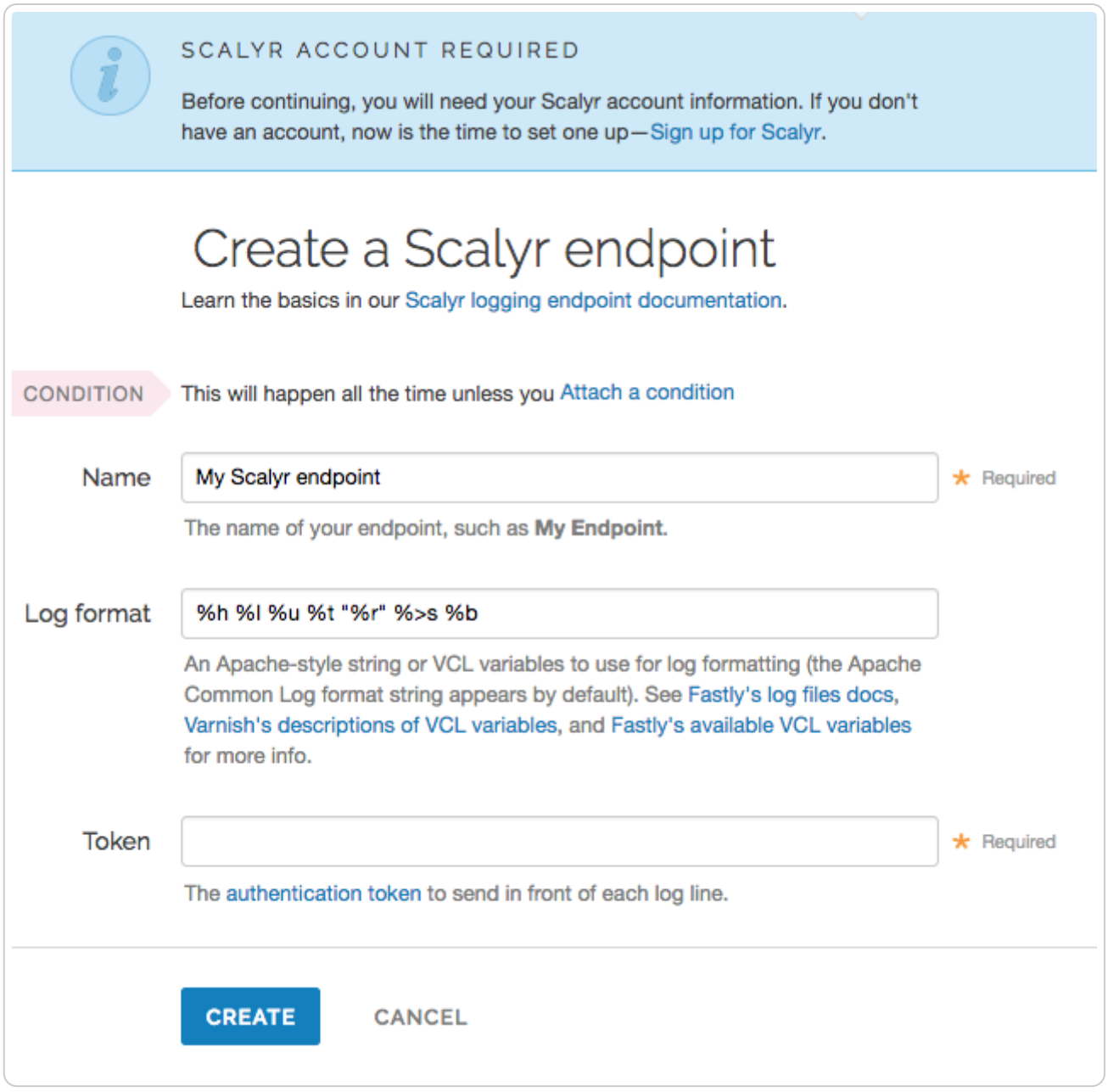
Once you've signed up, navigate to the **API Keys** area in the **Settings** on your Scalyr dashboard and make note of your Scalyr Write Token. Scaylr uses this to associate data you send them with your account. You'll need this token when you set up your endpoint with Fastly.

If you're adding the Scalyr endpoint via the command line, instead of the web interface, you should also have your Fastly API token (/guides/account-management-and-security/using-api-tokens) and the service ID (/guides/account-management-and-security/finding-and-managing-your-account-info#finding-your-service-id) and version number of the Fastly service for which you'll be enabling Scalyr logging.

Adding Scalyr as a logging endpoint

Follow these instructions to add Scalyr as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Scalyr logo. The Create a Scalyr endpoint page appears.



3. Fill out the **Create a Scalyr endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings (/guides/streaming-logs/custom-log-formats) for more information.

- In the **Token** field, type the Scalyr Write Token provided in the Scalyr dashboard.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

---

§ Log streaming: Sumo Logic (/guides/streaming-logs/log-streaming-sumologic)

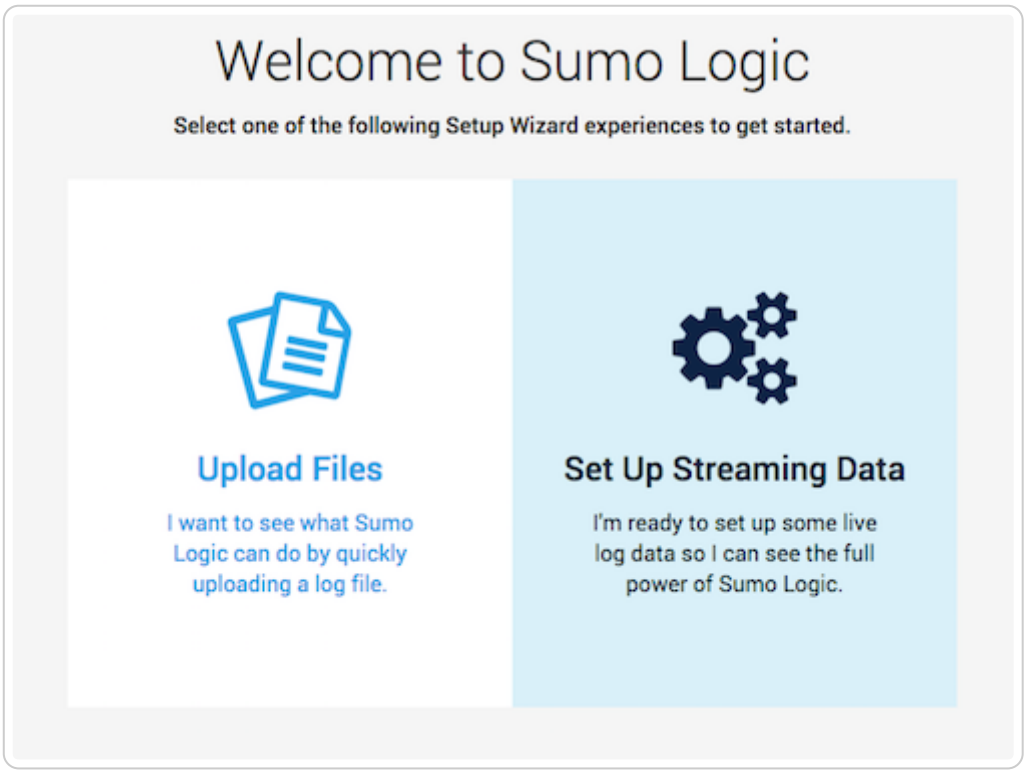
Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to Sumo Logic (https://www.sumologic.com/). Sumo Logic is a web-based log analytics platform used by developers and IT teams.

> ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

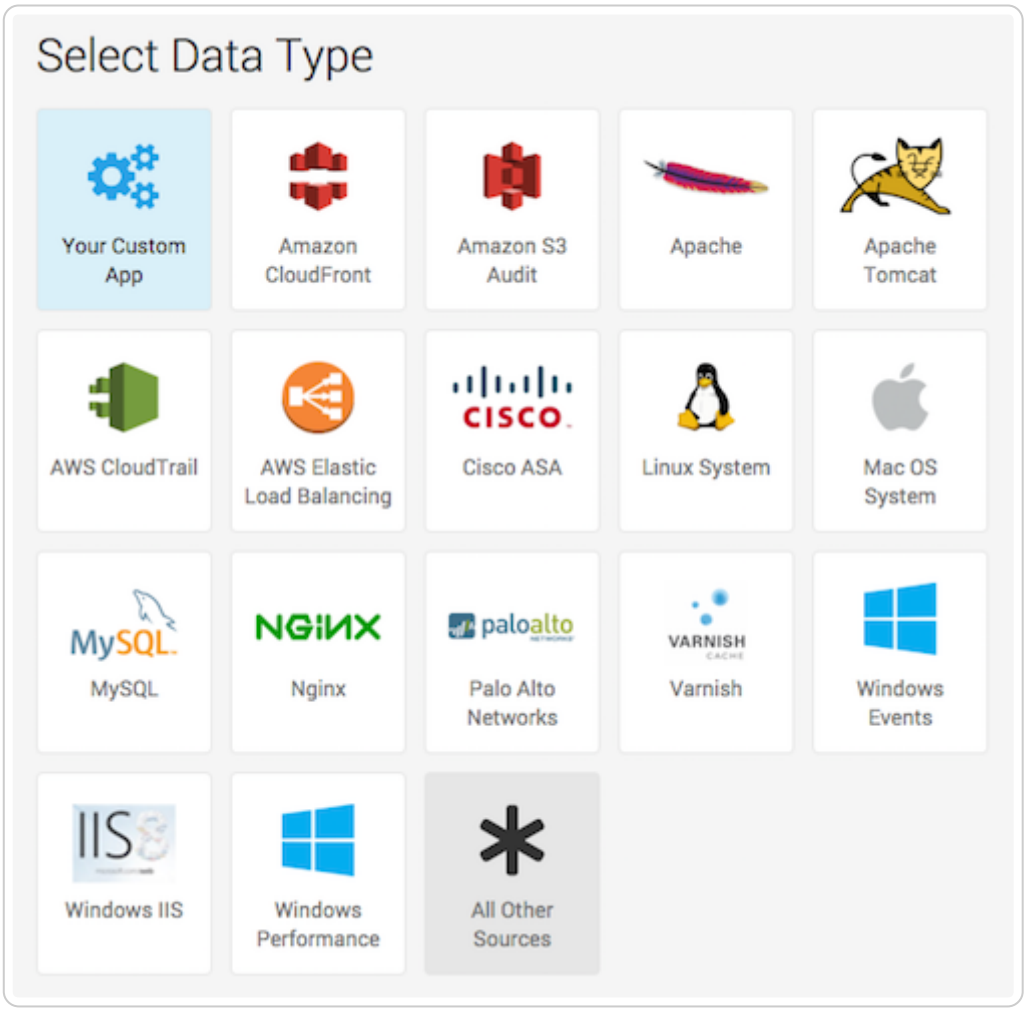
Setting up Sumo Logic

To use Sumo Logic as a logging endpoint, you'll need to create a Sumo Logic account, add a new source, and save the HTTP Source URL. Follow these instructions to add a new source in the Sumo Logic website:

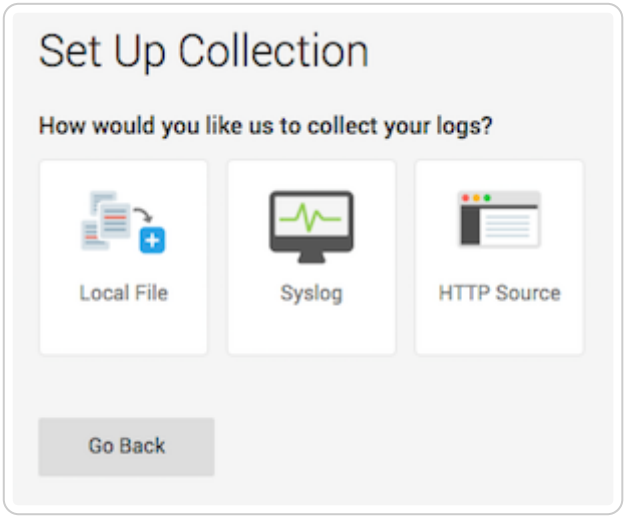
1. The process starts with the Sumo Logic Setup Wizard, which appears immediately after you create your Sumo Logic account. If you already have an account, you can access the wizard by selecting **Setup Wizard** from the **Manage** menu at the top of the Sumo Logic application.



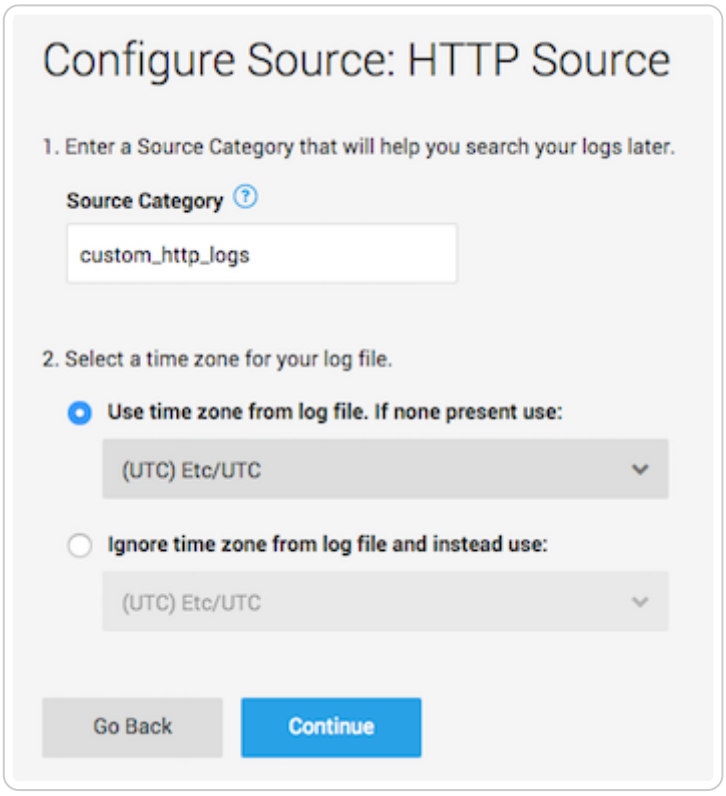
2. Click **Set Up Streaming Data**. The Select Data Type window appears.



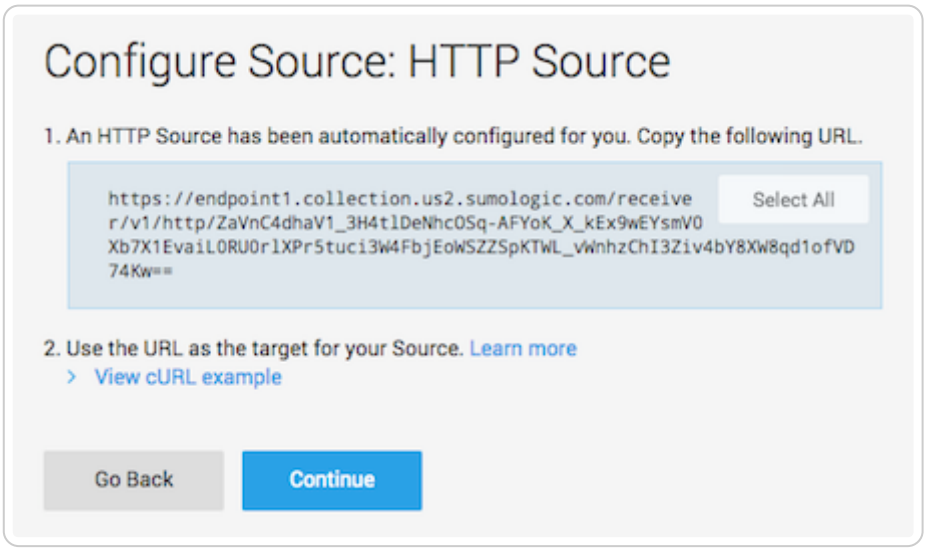
3. Click **All Other Sources**. The Set Up Collection window appears.



4. Click **HTTP Source**. The Configure Source: HTTP Source window appears.



5. In the **Source Category** field, type a human-readable name for the category (e.g., `fastly_cdn`) and select a time zone for your log file.
6. Click **Continue**. The HTTP Source URL appears.

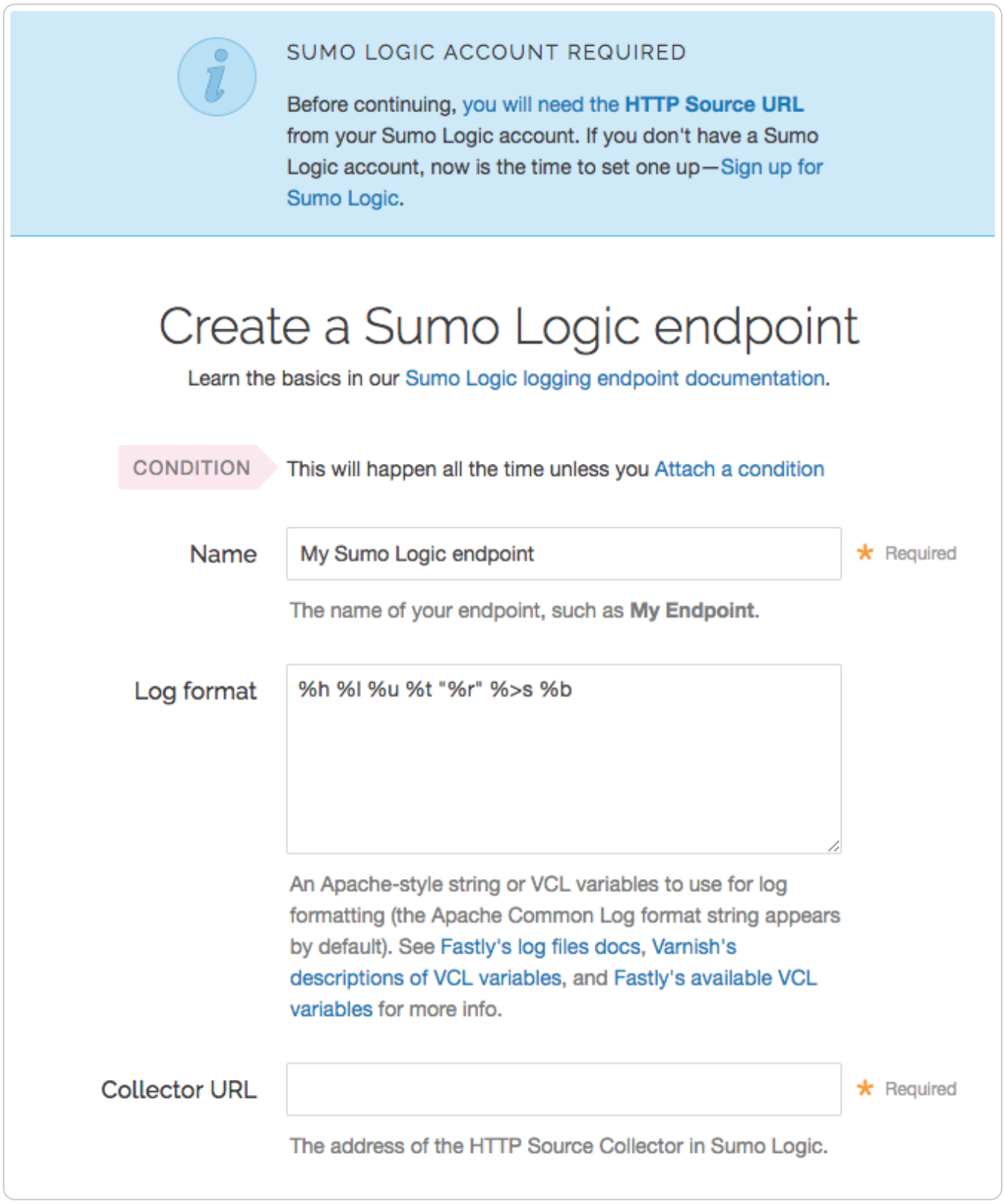


7. Copy the HTTP Source URL. You will enter this value in the Fastly web interface.
8. Click **Continue**. Sumo Logic will add the new source.

Adding Sumo Logic as a logging endpoint

After you've created a Sumo Logic account and obtained the HTTP Source URL, follow these instructions to add Sumo Logic as a logging endpoint for Fastly services:

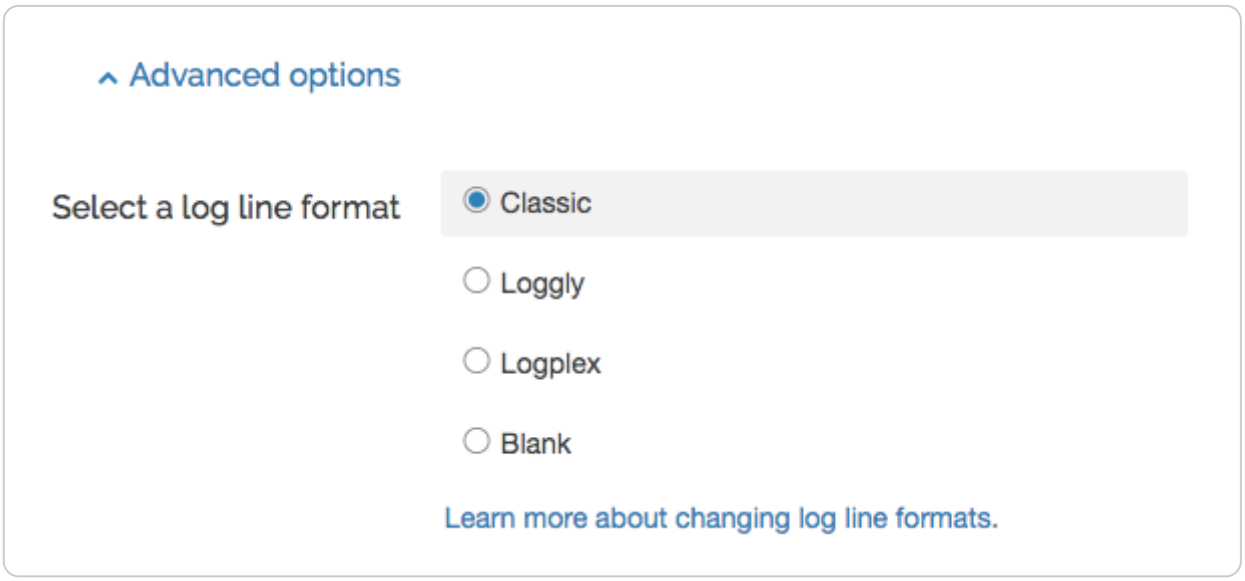
1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the Sumo Logic logo. The Create a Sumo Logic endpoint page appears.



3. Fill out the **Create a Sumo Logic endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. Our discussion of format strings (/guides/streaming-logs/custom-log-formats) provides more information.

- In the **Collector URL** field, type the address of the HTTP Source URL you found in the Sumo Logic website.

4. Click the **Advanced options** link of the **Create a Sumo Logic endpoint** page and decide which of the optional fields to change, if any.



5. Fill out the **Advanced options** of the **Create a Sumo Logic endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats) provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Troubleshooting

The Sumo Logic logging endpoint is designed for services with sustained levels of traffic. If you aren't seeing any logs in Sumo Logic, try waiting a bit.

---

§ Log streaming: Syslog (/guides/streaming-logs/log-streaming-syslog)

Fastly's Real-Time Log Streaming (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) feature can send log files to syslog-based logging software. Syslog is a widely used standard for message logging.

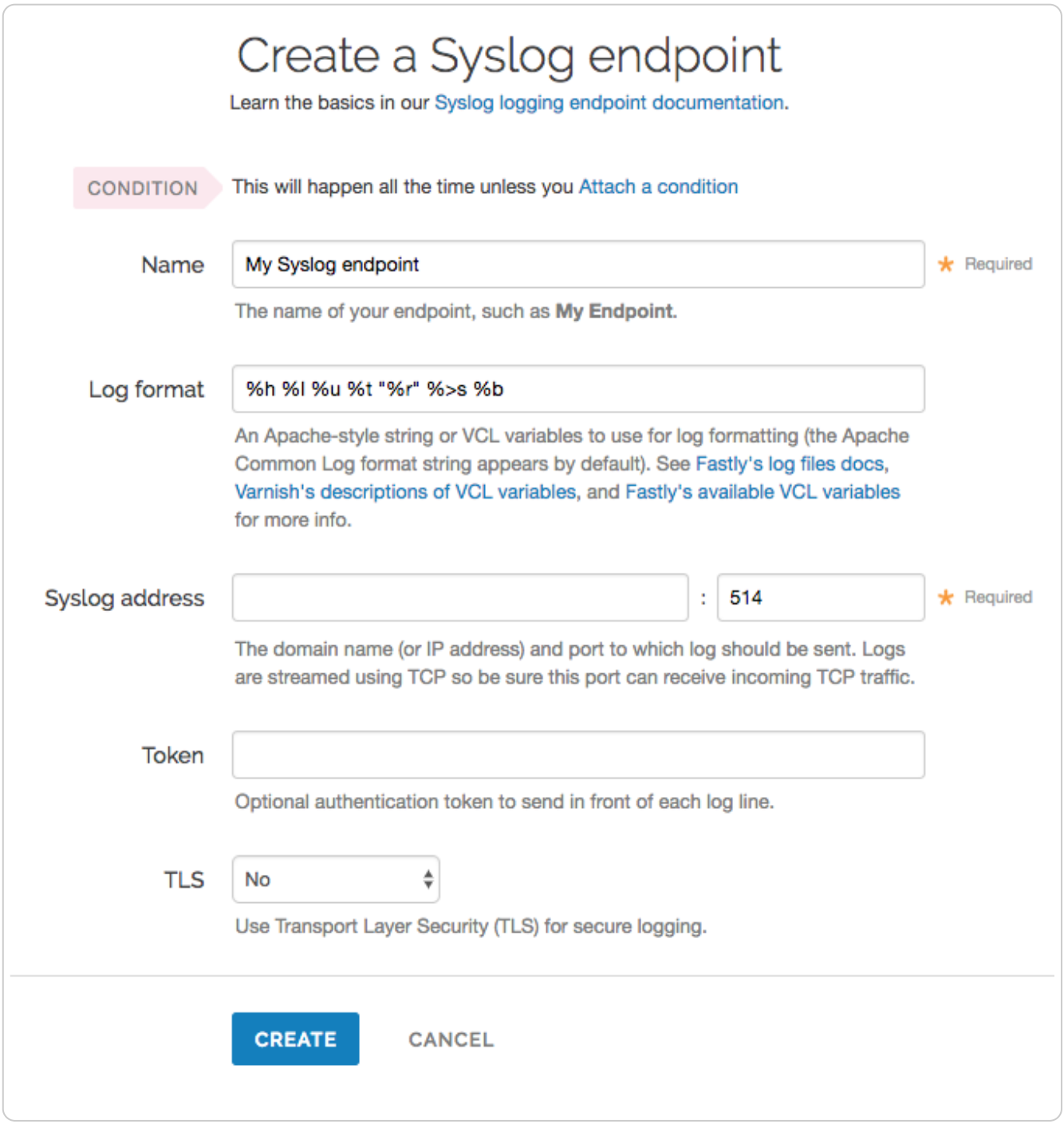
NOTE: Splunk Storm reached end-of-life status on April 1st, 2015. If you are using a Splunk (http://www.splunk.com/) product, you should configure it to receive log streams (/guides/streaming-logs/setting-up-remote-log-streaming) over syslog.

NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Adding syslog as a logging endpoint

Follow these instructions to add syslog as a logging endpoint:

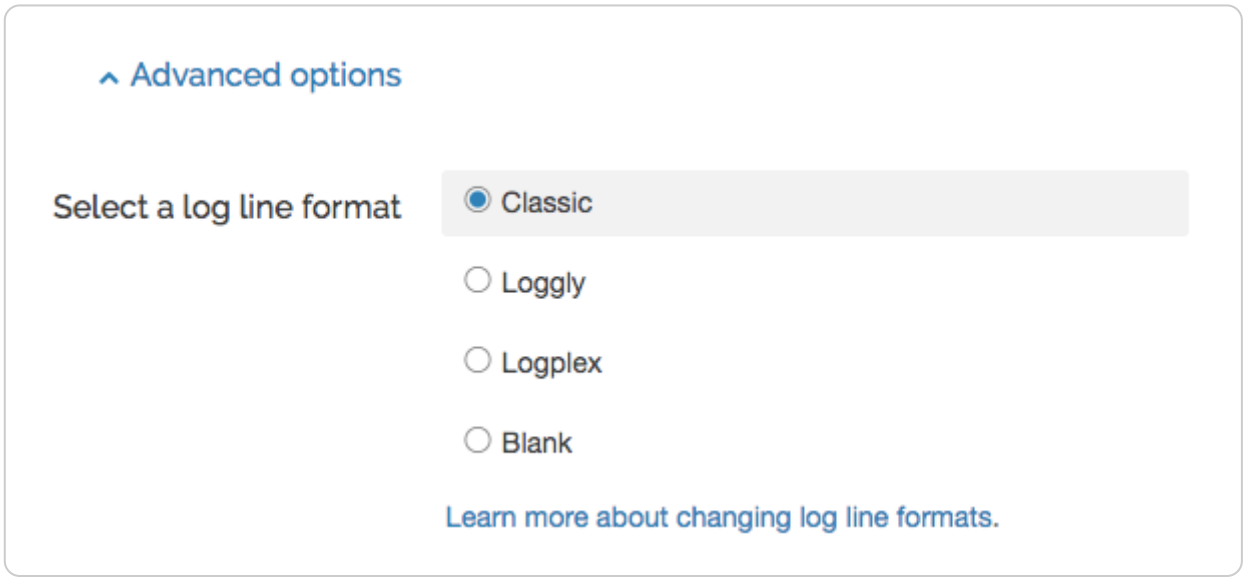
1. Review the information in our Setting Up Remote Log Streaming (/guides/streaming-logs/setting-up-remote-log-streaming) guide.
2. Click the syslog icon. The Create a Syslog endpoint page appears.



3. Fill out the **Create a Syslog endpoint** fields as follows:
    - In the **Name** field, type a human-readable name for the endpoint.
    - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings (/guides/streaming-logs/custom-log-formats) for more information.
    - In the **Syslog address** field, type the domain name or IP address and port to which logs should be sent. Be sure this port can receive incoming TCP traffic from Fastly. See the firewall considerations section for more information.

- In the **Token** field, optionally type a string prefix (line prefix) to send in front of each log line.
- From the **TLS** menu, select **No** to disable encryption for the syslog endpoint, or **Yes** to enable it. When you select Yes, the TLS Hostname and TLS CA Certificate fields both appear.
- In the **TLS Hostname** field, optionally type the hostname used to verify the syslog server's certificate. This can be either the Common Name (CN) or Subject Alternate Name (SAN). This field only appears when you select Yes from the Use TLS menu.
- In the **TLS CA certificate** field, optionally copy and paste the Certificate Authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certificate authority. This value is not required if your TLS certificate is signed by a well-known authority. This field only appears when you select Yes from the Use TLS menu.

4. Click the **Advanced options** link of the **Create a Syslog endpoint** page and decide which of the optional fields to change, if any.



5. Fill out the **Advanced options** of the **Create a Syslog endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats (/guides/streaming-logs/changing-log-line-formats) provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Adding separators or static strings

To insert a separator or other arbitrary string into the syslog endpoint format:

1. Create a new header (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) with the following fields:
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type any suitable header name (for example, `http.X-Separator`).
   - In the **Source** field, type any special character or string you want (for example, `"|"`).
2. Reference the new header variable in the log format box for your specific provider (for example, `req.http.X-Separator`).

Syslog facility and severity

The syslog output includes the following facility and severity values:

```
facility: local0
severity: info
```

Firewall considerations

Syslog has limited security features. For this reason, it's best to create a firewall for your syslog server and only accept TCP traffic on your configured port from our address blocks. Our list of address blocks is dynamic, so we recommend programmatically obtaining the list from our JSON feed (https://api.fastly.com/public-ip-list) whenever possible.

---

§ Setting up remote log streaming (/guides/streaming-logs/setting-up-remote-log-streaming)

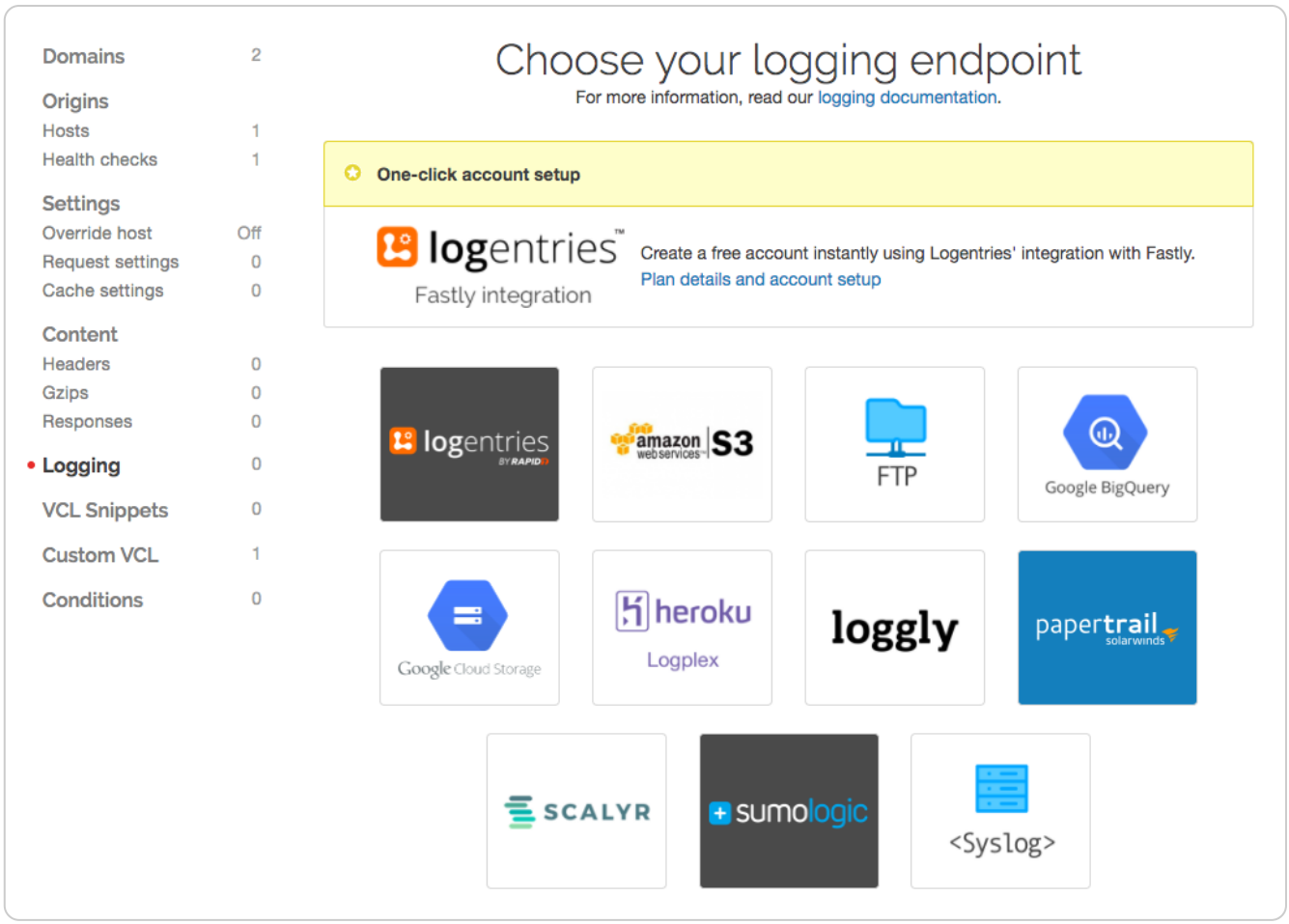
Fastly's Real-Time Log Streaming feature (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features) allows you to automatically save logs to a third-party service for storage and analysis. Logs provide an important resource for troubleshooting connectivity problems, pinpointing configuration areas that could use performance tuning (/guides/performance-tuning/), and identifying the causes of service disruptions. We recommend setting up remote log streaming when you start using Fastly services.

> ℹ **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service (https://www.fastly.com/terms) for more information.

Configuring logging endpoints

You can configure one or more logging endpoints for Fastly services. Follow these instructions to access the logging settings:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Logging** link. The logging endpoints page appears.



> **NOTE:** Some third-party services are disabled by default, so you won't see them in the Fastly web interface until they've specifically been enabled for your account. To enable these services, contact support@fastly.com (mailto:support@fastly.com).

5. Follow the instructions in one of our guides for third-party services to complete the set up process and deploy your changes:
   - Amazon S3 (/guides/streaming-logs/log-streaming-amazon-s3)
   - Cloud Files (/guides/streaming-logs/log-streaming-cloudfiles)
   - DigitalOcean Spaces (/guides/streaming-logs/log-streaming-digitalocean-spaces)
   - FTP (/guides/streaming-logs/log-streaming-ftp)

- Google BigQuery (/guides/streaming-logs/log-streaming-google-bigquery)
- Google Cloud Storage (/guides/streaming-logs/log-streaming-google-cloud-storage)
- Log Shuttle (/guides/streaming-logs/log-streaming-log-shuttle)
- Logentries (/guides/streaming-logs/log-streaming-logentries)
- Loggly (/guides/streaming-logs/log-streaming-loggly)
- Heroku's Logplex (/guides/streaming-logs/log-streaming-logplex)
- Papertrail (/guides/streaming-logs/log-streaming-papertrail)
- OpenStack (/guides/streaming-logs/log-streaming-openstack)
- Scalyr (/guides/streaming-logs/log-streaming-scalyr)
- Sumo Logic (/guides/streaming-logs/log-streaming-sumologic)
- Syslog (/guides/streaming-logs/log-streaming-syslog)

Once you've clicked Activate to deploy your changes, events will begin being logged immediately. The logs may take a few moments to appear on your log server.

How, when, and where logs are streamed

To control log streaming, Fastly provides two versions of custom log formats (/guides/streaming-logs/custom-log-formats), each of which uses Apache-style logging directives (http://httpd.apache.org/docs/2.4/mod/mod_log_config.html). The logging format strings in each of these versions are based on the Common Log Format (https://httpd.apache.org/docs/trunk/logs.html#common) (CLF).

Logs are streamed over TCP, not UDP, optionally using TLS for security with supported endpoints. Additionally, if you are using custom VCL (/guides/vcl/uploading-custom-vcl) be sure to include the `#FASTLY log` (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl) macro in your `vcl_log` handler.

By default, logs are placed in your root directory every hour using the file naming format `YYYY-mm-ddThh:mm:ss-<server id>`. You can change both the frequency and path of these files. Our guide on changing where log files are written (/guides/streaming-logs/changing-where-log-files-are-written) provides more information.

Fastly uses several different log-server aggregation points and each will send logs files, none of which contain duplicate entries. These log files are created as soon as streaming starts and they're written to over the entire time period you specify (or the default). Once that time has passed, the files aren't touched any more and the logging process creates a new batch of files.

Escaping characters in logs

Logs respond to VCL (/guides/vcl/) like any other object. For example, the following code can escape quotes from User-Agent your log stream:

```
log {"syslog serviceid endpointname :: "} {"""} cstr_escape(req.http.user-agent);
```

Preventing duplicate log entries when using custom VCL

If you use custom VCL (/guides/vcl/uploading-custom-vcl) commands for logging, you may notice duplicate entries in your logs. This happens because logs are being generated by both Fastly and the custom VCL logging commands. You can eliminate the duplicate entries by adding a condition that prevents Fastly from generating log entries. Follow these instructions to add the condition:

1. On the Logging endpoints page, click the **Attach a condition** link next to the appropriate logging service. The Add a condition window appears.
2. Click **Create a new response condition**. The Create a new response condition window appears.
3. Fill out the **Create a new response condition** window as follows:
   - In the **Name** field, type a human-readable name for the condition.
   - In the **Apply if** field, type `!req.url`. That condition will never be met, and is what prevents Fastly from generating log entries.
   - Leave the default value set in the **Priority** field.
4. Click **Save and apply to**.
5. Click the **Activate** button to deploy your configuration changes.

Fastly will stop generating log entries, and your logs will only contain entries generated by the custom VCL logging commands.

---

§ Useful conditions for logging (/guides/streaming-logs/useful-conditions-for-logging)

In addition to the standard logging directives (/guides/streaming-logs/custom-log-formats), the following conditions (/guides/conditions) can be used for logging when you set up remote log streaming (/guides/streaming-logs/setting-up-remote-log-streaming).

Logging errors only

You can log errors only if you want a general purpose log that catches everything and a more detailed log if there's an error:

```
fastly_info.state == "ERROR"
```

You can also log only 500 errors:

```
resp.status >= 500 && resp.status < 600
```

Logging only specific URLs using an Edge Dictionary

Using an Edge Dictionary (/guides/edge-dictionaries) (e.g., `urls_to_log`), you can log specific URLs having issues:

```
table.lookup(urls_to_log, req.url.path) == "log"
```

If a URL becomes a problem, you can start logging it by using the API (/guides/edge-dictionaries) to add the URL's path to the dictionary as a key with the value "log".

Logging samples

If you have a high-volume service, you might want to log only a proportion of requests by using the `randombool` VCL function in a condition. The following example will log only one percent of all requests:

```
randombool(1,100)
```

You could combine that with an Edge Dictionary (/guides/edge-dictionaries) to change the percentage of requests logged without having to deploy a new version of your service. The following example uses an Edge Dictionary named `service_variables`:

```
randombool(std.atoi(table.lookup(service_variables, "logging_percentage", 0)), 100)
```

In the example above, if the key `logging_percentage` doesn't exist, nothing will be logged.

Using !req.url to construct a log string in custom VCL

If you want to construct a log string in custom VCL, you can use the `!req.url` condition. This condition never evaluates to true, so nothing is sent to Fastly logging objects. Instead, it ensures that the log statement is generated in your custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl).

§ Useful variables to log (/guides/streaming-logs/useful-variables-to-log)

In addition to the standard logging directives (/guides/streaming-logs/custom-log-formats), the following request and response variables can be used for logging when you set up remote log streaming (/guides/streaming-logs/setting-up-remote-log-streaming). You can also log any Varnish variable (https://www.varnish-cache.org/docs/2.1/reference/vcl.html#variables). Consider taking advantage of some of Fastly's extensions to VCL (/guides/vcl/guide-to-vcl#fastlys-vcl-extensions) as well.

Time-related logging variables

These are the time-related variables that can be used for logging.

Variable	Description
`%{begin:%Y-%m-%dT%H:%M:%S%z}t`	The time of the start of the request in ISO 8601 format.
`%{end:%Y-%m-%dT%H:%M:%S%z}t`	The time of the end of the request in ISO 8601 format.
`%{time.elapsed.usec}V`	How long the request took in microseconds.
`%{time.start.sec}V`	When the request started in epoch seconds.

Connection-related logging variables

These are the connection-related variables that can be used for logging.

Variable	Description
`%{if(req.is_ipv6, "true", "false")}V`	Whether the request was over IPv6 or not.
`%{if(req.is_ssl, "true", "false")}V`	Whether the request was over HTTPS or not.
`%{cstr_escape(tls.client.protocol)}V`	Which version of TLS was used by the client.
`%{cstr_escape(tls.client.servername)}V`	Which SNI server name the client sent.
`%{cstr_escape(tls.client.cipher)}V`	Which cipher the TLS request used.
`%{cstr_escape(tls.client.ciphers_sha)}V`	Which cipher the TLS request used.

Variable	Description
`%{cstr_escape(tls.client.tlsexts_sha)}V`	A SHA of the TLS extension identifiers sent from the client as part of the TLS handshake, represented in base64.
`%{if(fastly_info.is_h2, "true", "false")}V`	Whether or not this was an HTTP2 request.
`%{if(fastly_info.h2.is_push, "true", "false")}V`	Whether or not this was an HTTP2 Push response.
`%{fastly_info.h2.stream_id}V`	What the HTTP2 Stream ID was.

Request- and response-related logging variables

These are the request- and response-related variables that can be used for logging.

Variable	Description
`%{Fastly-Orig-Host}i`	The original Host requested.
`%{Host}i`	The current Host request header (because it could have been modified to send to the origin).
`%{Referer}i`	The Referer request header. Specifically, which URL linked to this page.
`%{User-Agent}i`	The User-Agent request header. Specifically, which browser requested this page.
`%{Accept}i`	The Accept request header. Specifically, the types of content the client can accept.
`%{Accept-Language}i`	The Accept-Language request header. Specifically, the human languages the client can respond with.
`%{Accept-Encoding}i`	The Accept-Encoding request header. Specifically, the content encoding the client is able to understand.
`%{Accept-Charset}i`	The Accept-Charset request header. Specifically, the character set encodings the client accepts.
`%{Connection}i`	The Connection request header. Specifically, whether or not the client can do keep-alive connections.
`%{DNT}i`	The DNT request header. Specifically, whether or not the client is sending a "Do Not Track" header.
`%{Forwarded}i`	The Forwarded request header. Specifically, the originating IP address of a request if this request is proxied.

Variable	Description
`%{Via}i`	The Via request header. Specifically, the intermediate protocols and recipients between the user agent and the server on proxied requests.
`%{X-Requested-With}i`	The X-Requested-With request header. Generally used to identify Ajax requests that will send the value XMLHttpRequest.
`%{X-Requested-For}i`	The X-Requested-For request header. Specifically, the originating IP address of a request if this request is proxied.
`%{X-ATT-DeviceId}i`	The X-ATT-DeviceId request header. Specifically, the make, mode, or firmware of AT&T devices.
`%{Content-Type}o`	The Content-Type response header. Specifically, the MIME type of the content.
`%{TSV}o`	The TSV response header. Specifically, the Tracking Status Value suggested for sending in response to a DNT request.

Cache-related logging variables

These are the cache-related variables that can be used for logging.

Variable	Description
`%{If-Modified-Since}i`	The If-Modified-Since request header. Specifically, the server will send back the requested resource, with a 200 status, only if it has been last modified after the given date.
`%{If-None-Match}i`	The If-None-Match request header. Specifically, the server will send back the requested resource, with a 200 status, only if it doesn't have an ETag matching the given ones.
`%{Cache-Control}o`	The Cache-Control response header. Specifically, whether or not all caching mechanisms from server to client may cache this object in seconds.
`%{Age}o`	The Age response header. Specifically, the age the object has been in a proxy cache in seconds.
`%{Expires}o`	The Expires response header. Specifically, the date and time after which the response is considered stale in "HTTP-date" format as defined by RFC 7231 (https://tools.ietf.org/html/rfc7231).

Variable	Description
`%{Last-Modified}o`	The Last-Modified response header. Specifically, the last modified date for the requested object in "HTTP-date" format as defined by RFC 7231 (https://tools.ietf.org/html/rfc7231). Used in conjunction with the If-Modified-Since request header.
`%{ETag}o`	The ETag response header. Specifically, an identifier for a specific version of a resource. Used in conjunction with the If-None-Match Request header.
`%{obj.hits}V`	The number of hits this object has (cache specific).
`%{obj.lastuse}V`	The last time this object was used (cache specific).

And these Fastly-specific ones:

Variable	Description
`%{if(fastly_info.state ~"^(HIT|MISS)(?:-|$)", "true", "false")}V`	Whether this object is cacheable or not.
`%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIPE)).*", "\\2\\3") }V`	Whether the response was a HIT, MISS, PASS, ERROR, PIPE, HITPASS, or SYNTH(etic).

Geographic logging variables

These are the geographic variables that can be used for logging.

Variable	Description
`%{server.datacenter}V`	Which Fastly datacenter this request hit.
`%{client.geo.city}V`	Which city Fastly thinks the request originated from.
`%{client.geo.country_code}V`	Which country Fastly thinks the request originated from.
`%{client.geo.continent_code}V`	Which continent Fastly thinks the request originated from.
`%{client.geo.region}V`	Which region Fastly thinks the request originated from.

Size-related logging variables

These are the size-related variables that can be used for logging.

Variable	Description
`%{req.header_bytes_read}V`	The size of the request headers.
`%{req.body_bytes_read}V`	The size of the request body.
`%{resp.header_bytes_written}V`	The size of the response headers.
`%{resp.body_bytes_written}V`	The size of the response body.

Socket-related logging variables

These are the socket-related variables that can be used for logging.

Variable	Description
`%{client.socket.cwnd}V`	The client socket congestion window.
`%{client.socket.nexthop}V`	The IP address of the next gateway.
`%{client.socket.tcpi_rcv_mss}V`	The client socket max segment size for receiving.
`%{client.socket.tcpi_snd_mss}V`	The client socket max segment size for sending.
`%{client.socket.tcpi_rtt}V`	The client socket smoothed round-trip time in microseconds.
`%{client.socket.tcpi_rttvar}V`	The client socket round-trip time variance in microseconds.
`%{client.socket.tcpi_rcv_rtt}V`	The client socket round-trip time variance in microseconds for receives.
`%{client.socket.tcpi_rcv_space}V`	The current buffer space available for receiving data.
`%{client.socket.tcpi_last_data_sent}V`	The time since last data sent on client socket in microseconds.
`%{client.socket.tcpi_total_retrans}V`	The total number of packet retransmissions on the client socket.
`%{client.socket.tcpi_delta_retrans}V`	The change in number of packet retransmissions on the client socket.
`%{client.socket.ploss}V`	The client socket packet loss.

Miscellaneous logging variables

These are the miscellaneous variables that can be used for logging.

Variable	Description
`%{LF}V`	Literal new line (i.e., `"\n"`)

§ Browser recommendations when using the Fastly web interface (/guides/debugging/browser-recommendations-when-using-the-fastly-web-interface)

We support the latest version of the following browsers:

- Google Chrome (http://www.google.com/chrome)
- Firefox (https://www.mozilla.org/en-US/firefox/products/)
- Safari (http://www.apple.com/safari/download/)

If you aren't using one of these browsers, then some visual styling may not be correct when using the Fastly web interface (https://manage.fastly.com/).

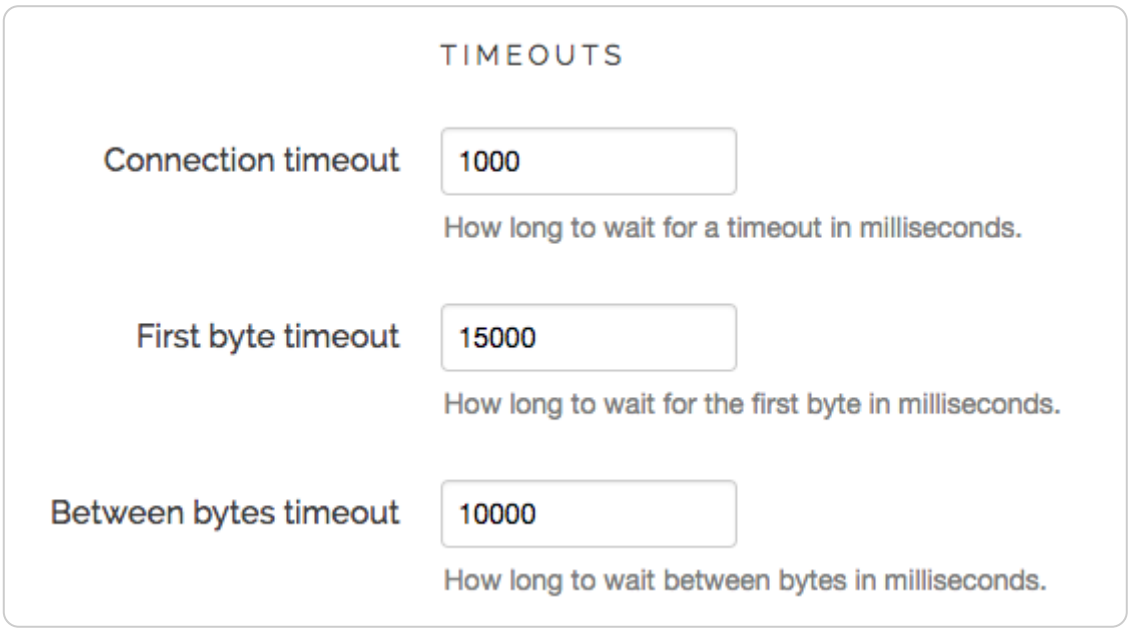
We strongly recommend updating your browser before beginning any debugging (/guides/debugging/) of Fastly services and before reporting problems to Fastly Customer Support (mailto:support@fastly.com). You can find the latest, downloadable versions of all major browsers online. The list at Browse Happy (http://browsehappy.com/) may help you.

§ Changing connection timeouts to your origin (/guides/debugging/changing-connection-timeouts-to-your-origin)

Connection timeouts to your origin server control how long Fastly will wait for a response from your origin server before exiting with an error. Changing the connection timeout is a good way to start troubleshooting 503 backend read errors (/guides/debugging/common-503-errors#error-503-

backend-read-error). Follow the steps below to change the connection timeouts to your origin server:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the link of the host that you want to edit. The Edit a host page appears.
6. Click the **Advanced options** link.



7. Type the new timeout in the appropriate field of the **Timeouts** section.
8. Click the **Update** button.

★ **TIP:** Additional techniques that help you gain insights into your service configurations can be found in our Debugging (/guides/debugging/) guides.

§ Checking cache (/guides/debugging/checking-cache)

Checking the cache status of an object on your website can help when troubleshooting problems. You can use the web interface or the cURL command to check Fastly's cache nodes for a cached object, and you can use the information provided to examine the objects's status, response time, and content hash.

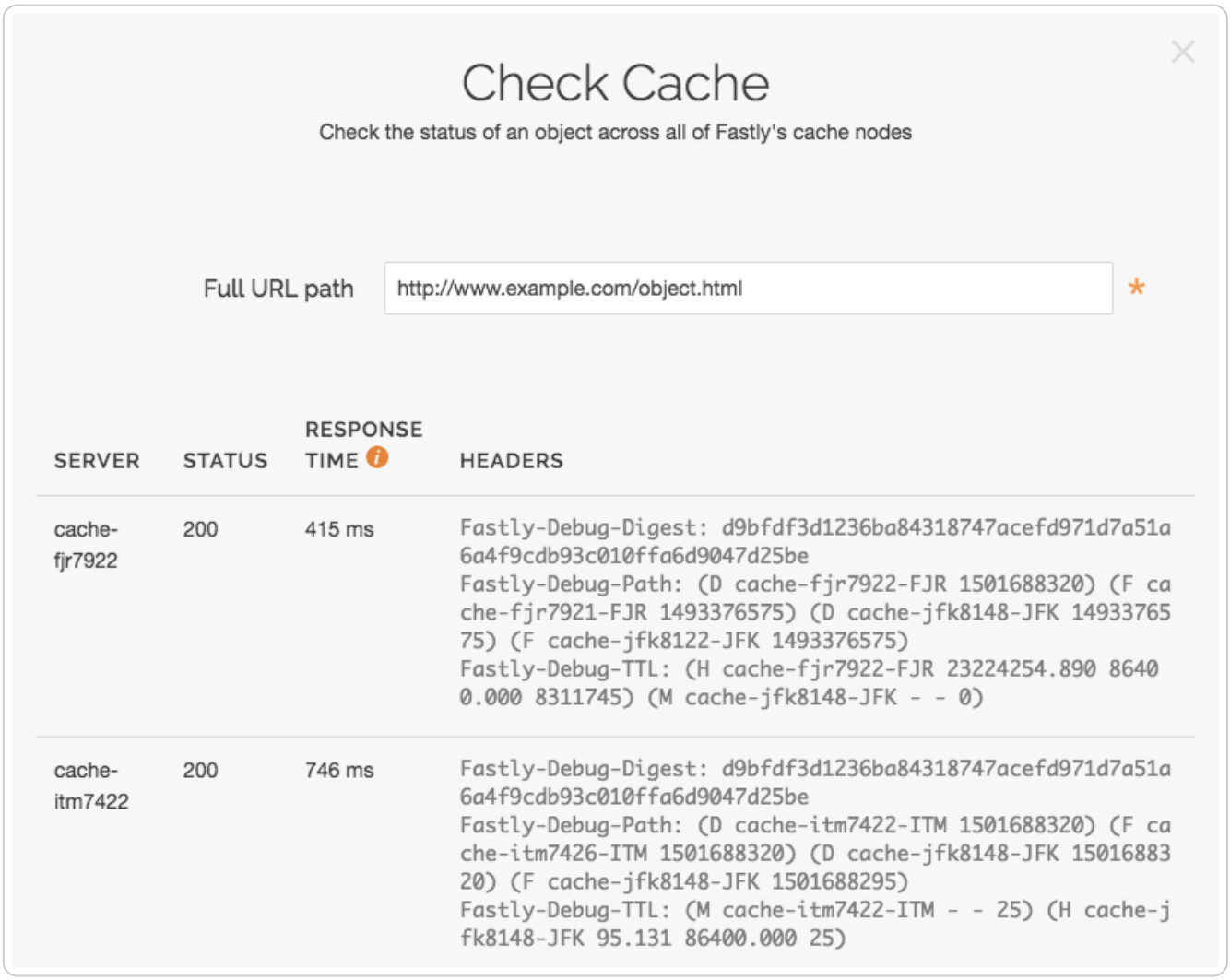
Using the web interface

Follow the steps below to check the cache status of an object using the Fastly web interface:

1. Log in to the Fastly web interface and click the **Configure** link.
2. Click the **Check Cache** button. The Check Cache window appears.



3. In the **Full URL path** field, type the full path to the object (e.g., `http://www.example.com/object.html`).
4. Click the **Check Cache** button. The results are displayed in the Check Cache window.



You can use this information to verify that the same copy of an object is stored on all of our servers. If the content hash (`Fastly-Debug-Digest`) is different across nodes, that usually indicates that there's a caching problem.

Using cURL

The easiest way to tell if your request is caching in the Fastly network is to use the check cache feature in the Fastly web interface, but if you prefer command line utilities, you can also use cURL (/guides/basic-setup/glossary-of-terms#curl). We recommend using one of two cURL commands for debugging purposes:

- a simple cURL command that displays the request and response headers for a given object
- a slightly more complex cURL command that uses the `Fastly-Debug` header to expose information normally stripped by the simple cURL

Using the simple cURL command

The following cURL command displays the request and response headers for a given object:

```
curl -svo /dev/null www.example.com/index.html
```

where `www.example.com/index.html` is replaced with the full object path of the object you're testing.

For example, using `curl -svo /dev/null www.example.com` produces something like the following section of output:

```
[...]

< Age: 142
< X-Served-By: cache-jfk1041-JFK, cache-ord1720-ORD
< X-Cache: HIT, HIT
< X-Cache-Hits: 1, 7

[...]
```

This output tells us the current age of the object in cache. It also shows shielding is enabled because two cache nodes display in `X-Served-By`. However, we're most interested in the output of the `X-Cache` header. A properly caching object displays a value of `X-Cache: HIT`, `X-Cache: HIT, HIT`, `X-Cache: HIT, MISS`, or `X-Cache: MISS, HIT`.

Using a Fastly-Debug header with cURL

The `Fastly-Debug` header provides additional information for debugging by exposing specific information that is normally stripped when using a simple cURL command:

`curl -svo /dev/null -H "Fastly-Debug:1" www.example.com/index.html`

where `www.example.com/index.html` is replaced with the full object path of the object you're testing.

For example, with optional shielding being used and a TTL set to 86400 (24 hours) using `Surrogate-Control`, the command `curl -svo /dev/null -H "Fastly-Debug:1" www.example.com` produces something like the following section of output:

```
[...]

< Surrogate-Control: max-age=86400
< Surrogate-Key: articles articles/1 articles/2

[...]
< Age: 403
< Fastly-Debug-Path: (D cache-ord1722-ORD 1470672957) (F cache-ord1743-ORD 1470672629) (D cache
-jfk1041-JFK 1470672629) (F cache-jfk1030-JFK 1470672554)
< Fastly-Debug-TTL: (H cache-ord1722-ORD 85997.246 0.000 403) (H cache-jfk1041-JFK - - 75)
< X-Served-By: cache-jfk1041-JFK, cache-ord1722-ORD
< X-Cache: HIT, HIT
< X-Cache-Hits: 1, 6

[...]
```

Because surrogate keys are present, the `Fastly-Debug` header exposes them. As with the simple cURL command, this section of output tells us the current age of the object in cache. In addition, `Fastly-Debug` exposes specific header details to help with debugging as noted below.

Information exposed by the Fastly-Debug header

Fastly-Debug Path contains information about which cache server handles fetching and delivery of an object and object TTL information. The edge POP appears first in the sequence and the shield POP appears second.

- `D` represents which cache by name in the edge or shield ran `vcl_deliver`
- `F` represents which cache by name in the edge or shield ran `vcl_fetch`
- the number following each specific server name is a timestamp in seconds

With shielding enabled, you should generally see four cache servers listed in this header. In rare cases where a cache server exists as both an edge and a shield within the cluster for that object, you may see two or three caches listed.

Fastly-Debug-TTL provides information on HIT and MISS timings.

- `H` represents a HIT, meaning the object was found in the cache
- `M` represents a MISS, meaning the object was not cached at the time of the query

For each of these timings:

- the first number specifies the TTL remaining for the object
- the second number specifies the grace period
- the third number specifies the current age of the object in cache

Because the cluster node determines these values, it may take a few requests to see these numbers populate as expected.

X-Served-By indicates the shield and edge servers that were queried for the request. The shield POP appears first in the sequence and the edge POP appears second.

X-Cache indicates whether the request was a HIT or a MISS for the datacenter.

> ℹ **NOTE:** Our guide to understanding cache HIT and MISS headers (/guides/performance-tuning/understanding-cache-hit-and-miss-headers-with-shielded-services) provides in depth details key to understanding the `X-Served-By`, `X-Cache`, and `X-Cache-Hits` headers with shielded services.

---

§ Common 503 errors (/guides/debugging/common-503-errors)

Varnish (/guides/vcl/guide-to-vcl), the software that powers the Fastly CDN, will sometimes return standardized 503 responses due to various issues that can occur when attempting to fetch data from your origin servers. The generic status text associated with a 503 error is "Service Unavailable." It can mean a wide variety of things. The most common reasons this generic text appears include:

1. The origin server generated a 503 error and Fastly passed it through as is.
2. The origin returned a 503 error without a response header, so Fastly used the default response.
3. The status line of the HTTP response from the origin was not parseable.
4. VCL code was run that used the "error" statement without an appropriate response status (e.g., `error 503` instead of `error 503 "_broken thing_"`).

The following list provides the most common non-generic, standardized 503 responses and basic explanations for each.

> ⚠ **WARNING:** If you are seeing 503 errors, do not purge all (/guides/purging/single-purges#purging-all-content) cached content. Purge all overrides stale-if-error (/guides/performance-tuning/serving-stale-content) and increases the requests to your origin server, which could result in additional 503 errors.

Timeout errors

The following describes typical timeout errors you may encounter.

Error 503 backend read error

This error typically appears if a timeout error occurs when Fastly cache servers attempt to fetch content from your origins. It can also be due to a variety of transient network issues that might cause a read to fail (e.g., router failovers, packet loss) or an origin overload.

Benchmarking your backend response times. Many outside factors cause backends response times to vary. Repeated, consistent backend read errors frequently can be prevented by changing your backend timeout settings in the Fastly web interface. Start by running the following command to estimate response time for benchmarking purposes:

```
curl -s -w "%{time_total}\n" -o /dev/null http://example.com/path/to/file
```

Increasing your backend timeout settings. After benchmarking some of the slower paths in your application, you should have an idea of your ideal backend response time. Adjust the backend timeout values on the Edit a host page in the Advanced options area (/guides/debugging/changing-connection-timeouts-to-your-origin). Also, if there is an external interface in front of the origin (such as a load balancer or firewall), review the timeouts for these interfaces.

Error 503 connection timed out

This error occurs if the request times out while waiting for Fastly to establish a TCP connection to your origin or waiting for your origin to respond to the request. Similar to backend read errors, connection timeouts can be caused by transient network issues, long trips to origin, and origin latency. Two common ways to alleviate these timeout errors include:

- Increasing the connection timeout values set for the Fastly host (/guides/debugging/changing-connection-timeouts-to-your-origin).
- Setting up an origin shield (/guides/performance-tuning/shielding). Setting up an origin shield provides two advantages:
  - Shortening the distance needed to establish a connection.
  - Reducing TCP handshakes resulting from using multiple POPs. This allows the origin to avoid slowdowns and to process only requests on a few connections from the shield.

Error 503 backend write error

This error is similar to the backend read error but occurs when Fastly sends information in the form of a POST request to the backend. This error can be resolved the same way as the backend read error.

Error 503 client read error

This error generally occurs because of a network issue between the client and Fastly. It can also occur when a user abandons the loading of a page (e.g., a page is loading too slowly and the user clicks stop in the browser). It is similar to the backend read error but occurs when reading information from a client. If you get this error, contact Fastly support (mailto:support@fastly.com) for help identifying the network issue.

Origin configuration errors

The following describes typical origin configuration errors you may encounter.

Error 503 connection refused

This error occurs when Fastly attempts to make a connection to your origin over a specific port and the server refuses the connection. It typically appears when the wrong port is specified for the host in the Fastly web interface. To resolve this error, you may need to adjust your port number (/guides/basic-configuration/connecting-to-origins) to ensure you're using the port needed to connect to your origin. If adjusting your port number doesn't work, you may also need review your origin configurations to ensure you're allowing connections from Fastly specific IPs (https://api.fastly.com/public-ip-list).

Error 503 illegal vary header from backend

This error occurs when a backend returns a malformed vary header with its response. A well-formed vary header (https://www.fastly.com/blog/best-practices-for-using-the-vary-header) tells Fastly to serve a different version of an object based on the value of the request header included within it.

Error 503 network unreachable

This error appears when Fastly can't find a route to the given IP range. This generally occurs because of misconfigured or non-operational routers. To resolve this error, check your routers to ensure they are operational or configured correctly.

Origin health errors

The following describes typical origin health errors you may encounter.

Error 503 backend is unhealthy

This error appears when custom health checks report a backend as down. It typically occurs when a Fastly edge server receives a client request and must make a request to your origin, but because the backend is considered unhealthy, Fastly doesn't try to send the request at all. This error may

mistakenly appear instead of the backend.max_conn reached error the first time Fastly encounters the maximum number of connections to your backend. Some of the reasons this error may occur are:

- the origin took too long to respond to the request
- there are transient network issues and the health check couldn't get to the origin
- the health check was misconfigured, or the resource the health check is checking against was removed or altered in some way

To resolve this error, check to make sure your origin is configured correctly and the object the health check is requesting exists at the specified location.

Error 503 no stale object available

This error occurs when you configure Fastly to serve stale objects (/guides/performance-tuning/serving-stale-content) in the event of a backend failure but the stale object has expired and your backend is still failing for some reason (thus, no stale object is available). To resolve this error, you will need either to fix your origin or check your network.

Connection limit errors

The following describes typical connection limit errors you may encounter.

Error 503 backend.max_conn reached

This error occurs when Varnish makes a request to a backend in your Fastly service that has reached its defined maximum number of connections. By default, Fastly limits you to 200 origin connections from a single edge node to protect the origins from overload. For the majority of sites, this should be enough. If you get this error message with less than 10,000 non-hit requests per second, make sure your origin is responding normally (e.g., there are no origin slow downs). If you just increase the number of maximum connections, you may be exacerbating the problem. If you have determined that your origin is not the issue, increase the maximum connections limit to your origin or reach out to Fastly support (mailto:support@fastly.com) for further help with this issue. This error may also appear as "Error 503 maximum threads for service reached."

Error 503 maximum threads for service reached

See Error 503 backend.max_conn reached.

Director errors

The following describes typical Director errors you may encounter.

Error 503 no healthy backends

This error occurs when a Director (/api/config#director) used for balancing requests among a group of backends (only available via the Fastly API) can't cache the specified content because there are no healthy backends available in its group.

Error 503 all backends failed or unhealthy

This error occurs when a Director (/api/config#director) used for balancing requests among a group of backends (only available via the Fastly API) fails because all the backends are unhealthy or multiple backends from which the Director tried to fetch information failed with the same error.

Error 503 quorum weight not reached

This error occurs when a Director (/api/config#director) used for balancing requests among a group of backends (only available via the Fastly API) can't serve traffic based on its configuration because it does not have enough available backends in its group.

To resolve any of these errors, you should either check for and resolve any issues with your origin or make sure the quorum setting is correct. Also, make sure you are setting the quorum setting correctly. For example, in a five backend director, 85% of the quorum will mark the director unhealthy if a single backend is unhealthy.

TLS errors

The following describes typical TLS errors you may encounter.

Error 503 SSL handshake error

This error occurs when TLS negotiation between Fastly and your origin fails. To fix this error, review and correct your host's TLS configurations (/guides/basic-configuration/connecting-to-origins#setting-the-tls-hostname).

Error 503 unable to get local issuer certificate

This error occurs when a certificate in the certificate chain (https://tools.ietf.org/html/rfc5280#section-3.2) is missing or invalid. To better determine which of these issues is the cause of the error, we suggest running an SSL test on your origin (https://www.ssllabs.com/ssltest/) to highlight any issues with the certificate installed there. There are two common ways you can resolve this error:

- For missing or invalid certificates, download and replace the missing or incorrect certificate.
- If both the intermediate and root certificates are correct, insert a valid Server Name Indication (SNI) hostname in the origin TLS options of your Fastly service.

Error 503 hostname doesn't match against certificate

This error occurs when the certificate hostname specified in your service's origin TLS settings does not match either the Common Name (CN) or available Subject Alternate Names (SANs). To resolve this error, enter a certificate hostname value that matches the CN or SAN entries on your origin's certificate.

Error 503:14077410:SSL routines:SSL23_GET_SERVER_HELLO:sslv3 alert

This error occurs when Server Name Indication (SNI) is required in the TLS handshake to origin, but the SNI hostname field is either blank or incorrect. To correct this error, enter a hostname value in the SNI hostname field. Often this will match the value specified in the certificate hostname field.

Error 503 certificate has expired

This error occurs when a certificate installed at the origin expires. To resolve this, renew your certificate or download a new one.

---

§ Common service and domain errors (/guides/debugging/common-service-and-domain-errors)

Exceeding max number of domains

We currently limit the maximum number of services and domains you can configure (including when you create domains programmatically (/guides/basic-configuration/working-with-domains)). Once you reach that limit, error messages may appear that look something like this:

```
{
  "msg": "An error occurred while connecting to the fastly API, please try your request again."
,
  "detail": "Exceeding max number of domains: 10"
}
```

If you're receiving a limit message and need to create more services or domains, contact support@fastly.com (mailto:support@fastly.com) for assistance. Fastly support engineers can not only increase the number of services that you can use, they can suggest other ways to design what you are trying to achieve.

---

§ Debugging with mtr (/guides/debugging/debugging-with-mtr)

For diagnostics and debugging in the Fastly network, we think the mtr (http://www.bitwizard.nl/mtr/) tool offers a great way to test network speed, evaluate performance (/guides/performance-tuning/), and perform connection diagnostics. The program's source and installation instructions live in GitHub (https://github.com/traviscross/mtr).

While mtr provides a number of practical uses for network engineering needs, the following command works well:

```
mtr -c 20 -w -r www.example.com
```

Be sure to replace `www.example.com` with the hostname of the domain you're working with. The command will generate the network hops to the destination you specify, any packet loss experienced, and aggregate connection statistics.

For example, if we wanted to test the network connection from Fastly's San Francisco office to the CDN, we would use the above command for `www.fastly.com`. The following output would appear:

```
~  mtr -c 20 -w -r www.fastly.com
Start: Mon Feb  2 15:27:20 2015
HOST: test-local-machine.local                Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.100.20.2                            0.0%    20    2.1   2.2   1.6   4.3   0.5
  2.|-- ge-4-3-4.mpr4.sfo7.us.zip.zayo.com     0.0%    20    2.3   2.4   1.8   5.2   0.6
  3.|-- ae5.cr2.sjc2.us.zip.zayo.com           0.0%    20    4.6   6.5   2.9  35.3   7.7
  4.|-- ae10.mpr4.sjc7.us.zip.zayo.com         0.0%    20    4.7   4.8   3.6  14.5   2.3
  5.|-- be6461.ccr21.sjc03.atlas.cogentco.com  5.0%    20    5.1   5.9   4.2  15.3   2.6
  6.|-- fastly-inc.edge2.sanjose3.level3.net   0.0%    20    5.0   4.7   4.2   8.2   0.8
  7.|-- ???                                   100.0    20    0.0   0.0   0.0   0.0   0.0
  8.|-- 23.235.47.184                          0.0%    20    4.7  14.3   3.8  74.6  20.3
```

§ Debugging with WebPageTest (/guides/debugging/debugging-with-webpagetest)

It's important to establish habits of testing and performance (/guides/performance-tuning/) before, during, and after migrating to Fastly. This allows you to clearly measure the impact of tests and changes to your infrastructure.

One tool that Fastly recommends for this purpose is WebPageTest.org (http://www.WebPageTest.org). WebPageTest provides a free and open source testing tool for deep performance analysis. It is built on browser technology to accurately replicate what your end users encounter when visiting a website.

We recommend using the WebPageTest defaults for basic testing, but keep a few rules in mind:

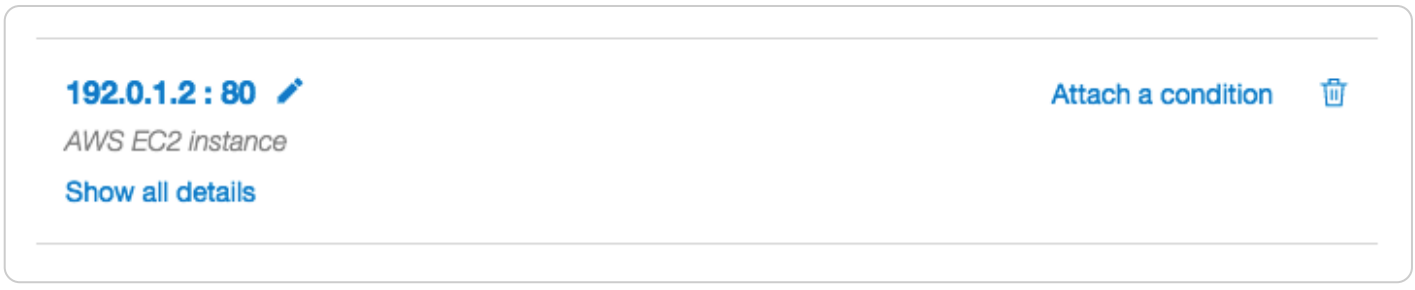
- On the Test Settings tab under Advanced Settings, Connection should always be set to Native Connection during initial benchmarks.
- Two to three test runs may be required before a site is properly caching in Fastly.
- Using WebPageTest's "Visual Comparison" (http://www.webpagetest.org/video/) feature offers an ideal way to A/B test potential changes.

§ Error 1000 with CloudFlare DNS (/guides/debugging/error-1000-with-cloudflare-dns)

Using CloudFlare for DNS and other CDNs can cause CloudFlare to show an Error 1000 indicating that your DNS points to prohibited IP addresses. This occurs when the hostnames are CNAMEed to Fastly (/guides/basic-setup/adding-cname-records) and an origin server is configured as a fully qualified domain name (FQDN) within Fastly:



To solve this error, direct Fastly to use the IP address as the host for any backend origin servers. This removes the need to resolve the hostname for traffic to the servers:



You can also change this by modifying the VCL configuration files directly. For example, this VCL:

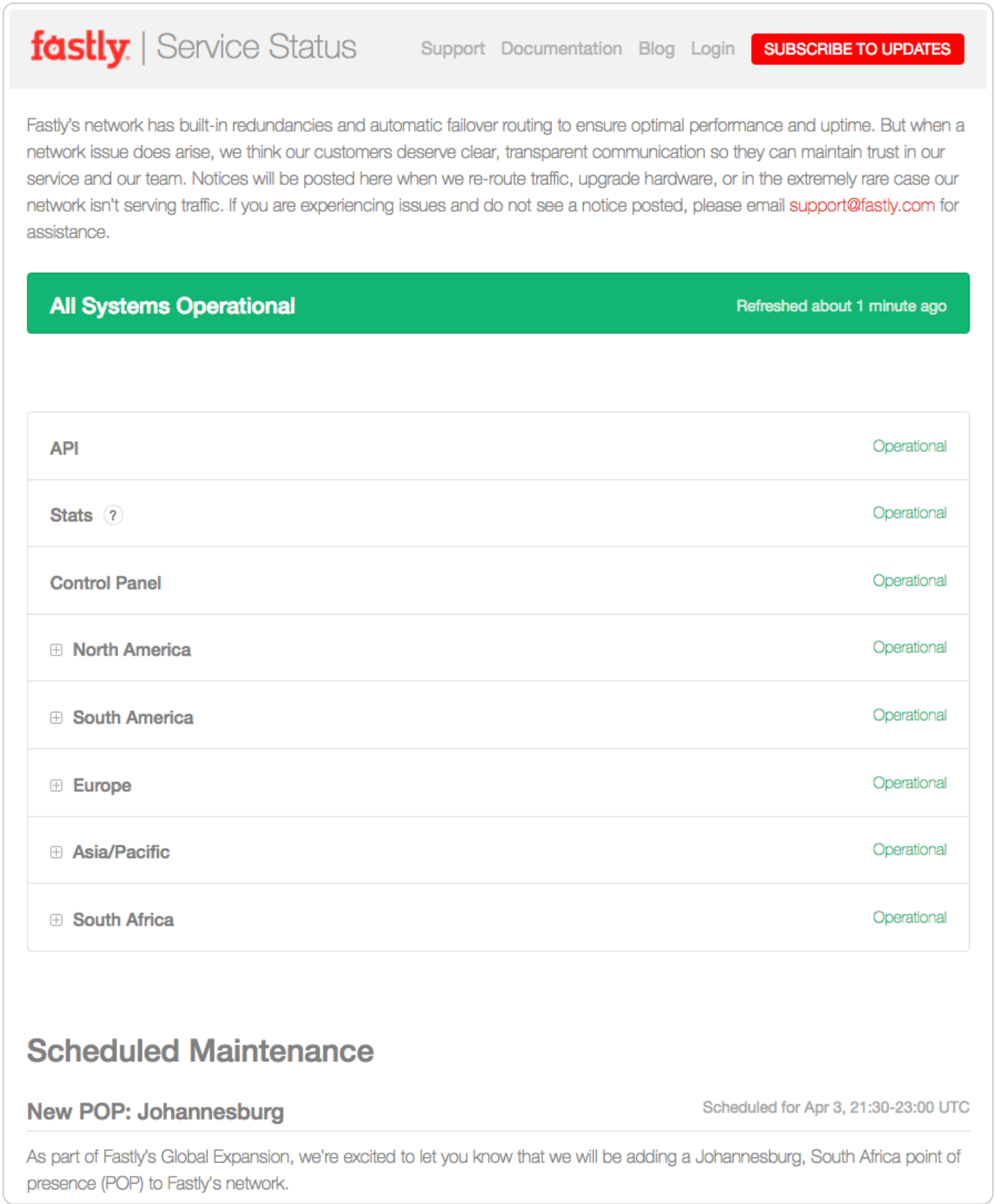
```
backend F_Hosting_server_Example_Backend {
  ...
  .port = "80";
  .host = "exampleserver.exampledomain.tld";
}
```

would become:

```
backend F_Hosting_server_Example_Backend {
  ...
  .port = "80";
  .host = "12.34.56.78";
}
```

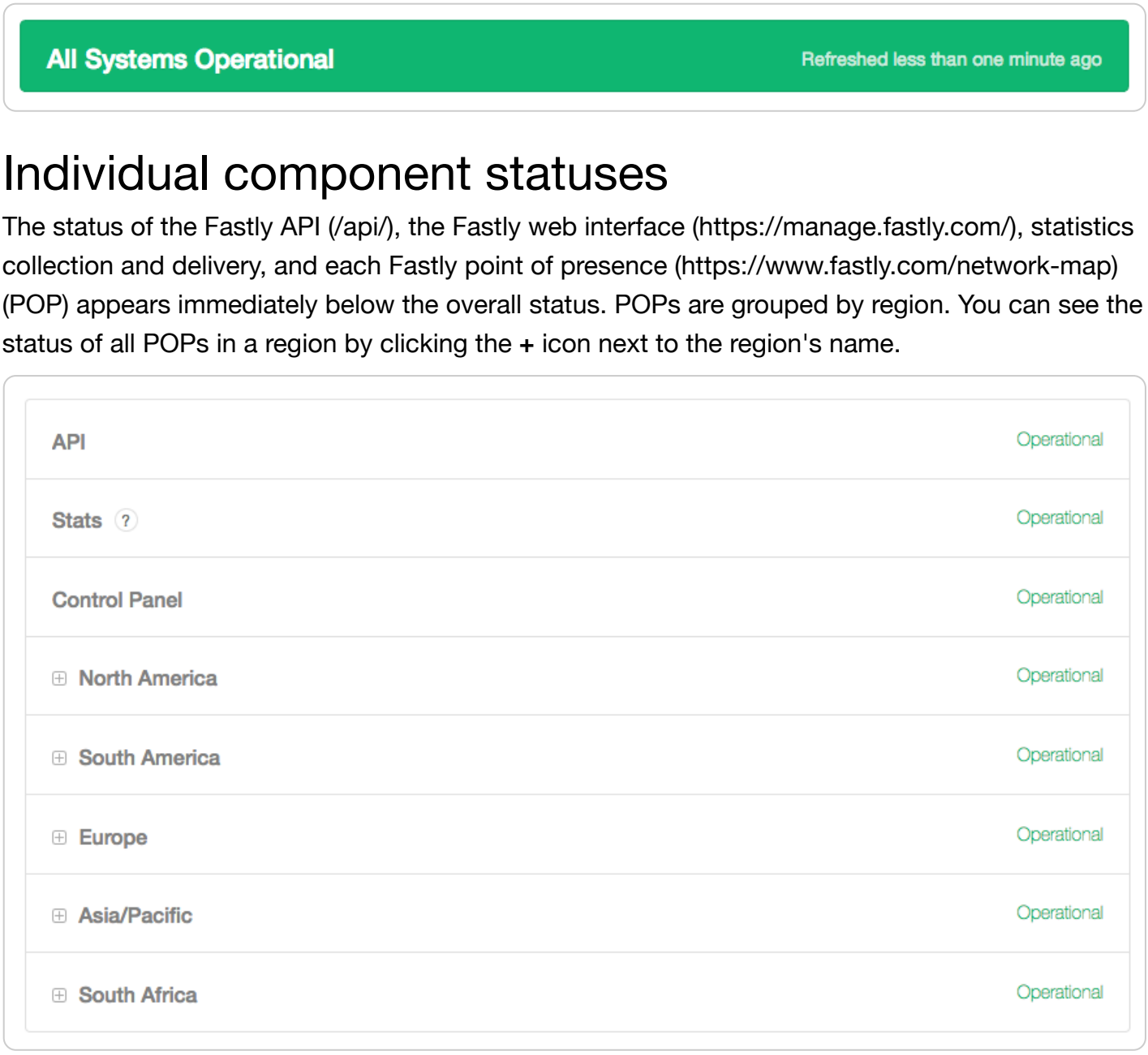
§ Fastly's network status (/guides/debugging/fastlys-network-status)

Fastly continuously monitors the status of our global network and all related services. In the event of a service interruption, an update will be posted on the Fastly status page at status.fastly.com (https://status.fastly.com). If you are experiencing problems and do not see a notice posted, email support@fastly.com (mailto:support@fastly.com) for assistance.



Overall system status

The current system status appears at the top of the Fastly status page and includes the last time the status was refreshed so that you know how current the information is.

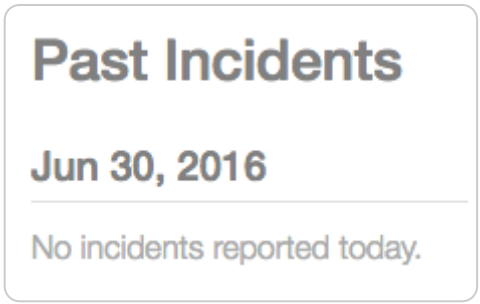


Individual component statuses

The status of the Fastly API (/api/), the Fastly web interface (https://manage.fastly.com/), statistics collection and delivery, and each Fastly point of presence (https://www.fastly.com/network-map) (POP) appears immediately below the overall status. POPs are grouped by region. You can see the status of all POPs in a region by clicking the **+** icon next to the region's name.

Past incident statuses

Fastly keeps track of past incidents. Past incidents, if any, for approximately the past two weeks appear immediately below the individual component statuses.



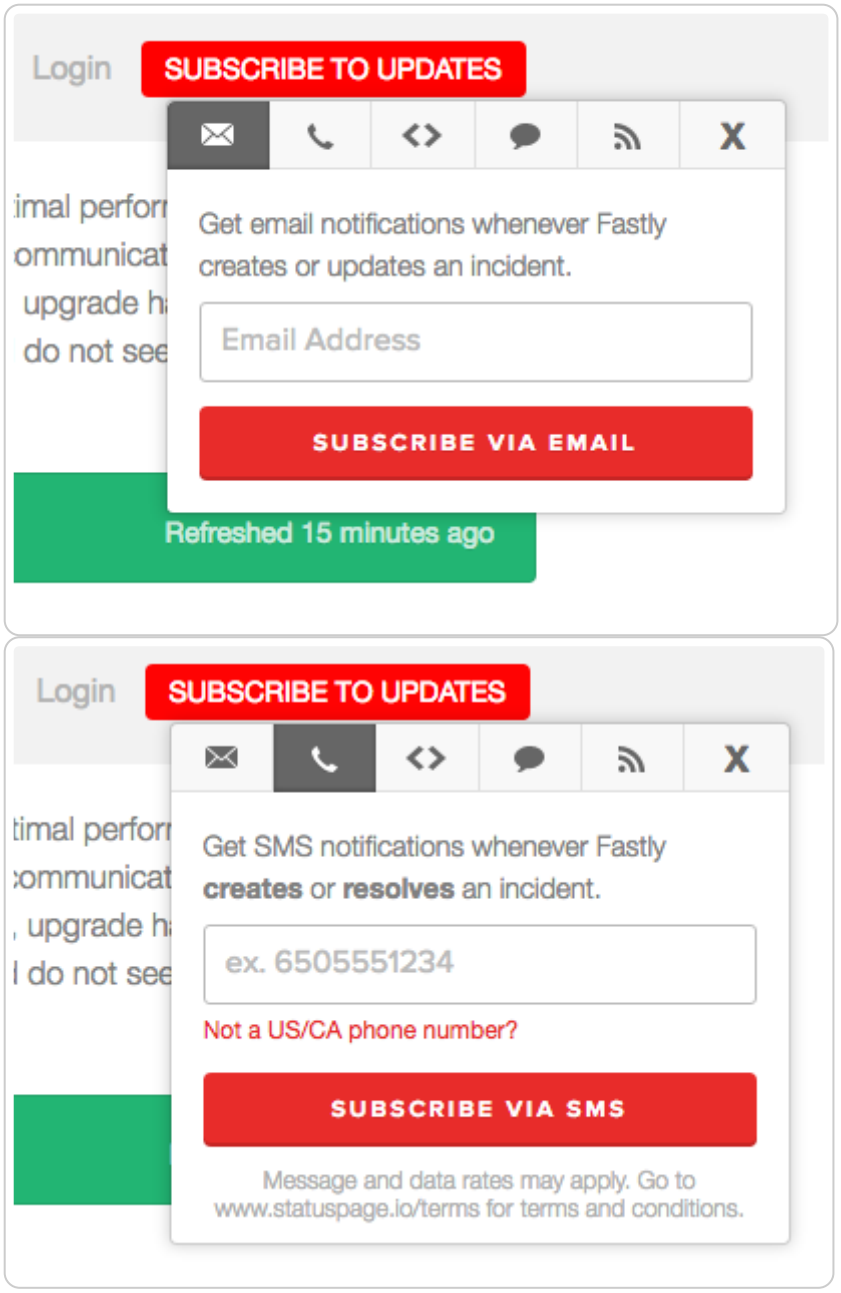
In addition to the textual description, each incident status appears in a color that indicates the level of service impact. The color indicators are as follows:

- Black - no component marked specifically as out of service or degraded
- Blue - scheduled maintenance
- Yellow - minor degradation or disruption
- Orange - significant degradation or traffic rerouting
- Red - component offline

We also keep track of all past incidents in an incident history page (https://status.fastly.com/history).

Subscribing to notifications

Fastly allows you to subscribe to status notifications via email or SMS text messaging. Simply click the **Subscribe to Updates** button in the upper right corner of the status page screen. Once subscribed, we'll email you any time we create or update an incident.



To subscribe to email notifications, click the letter icon, type your email address in the displayed field, and click **Subscribe Via Email**. You can unsubscribe at any time by clicking the unsubscribe link that appears at the bottom of every status email.

To subscribe via SMS text messaging, click the telephone icon, type your telephone number in the displayed field, and click **Subscribe Via SMS**. Unsubscribe from SMS text messaging at any time by replying STOP to any status message you receive.

---

§ Google Pagespeed module errors (/guides/debugging/google-pagespeed-module-errors)

If you are using the Google Pagespeed module and notice constant MISSes for HTML pages, check the Cache-Control settings (/guides/tutorials/cache-control-tutorial) in the module's .htcaccess file.

By default, Google Pagespeed serves all HTML with `Cache-Control: no-cache, max-age=0`. This setting conflicts with Fastly's default configuration. If your origin sends the headers `Cache-Control: private` or `Cache-Control: max-age=0`, Fastly will pass requests straight to the origin.

To change the Google Pagespeed directive and leave the original HTML caching headers, update your origin's .htcaccess file with:

```
ModPagespeedModifyCachingHeaders off
```

More details about the Pagespeed Module (https://developers.google.com/speed/pagespeed/module/) can be found within Google Developers directory. For additional information about controlling how long Fastly caches your resources, start with our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial)

---

§ Googlebot crawl stats (/guides/debugging/googlebot-crawl-stats)

Any time you notice any major changes in your SEO stats, indexing, or crawler behavior, start troubleshooting by asking these questions:

- Did you read the Google FAQs (https://sites.google.com/site/webmasterhelpforum/en/faq--crawling--indexing---ranking) for indexing, crawling, and ranking?
- Is your robots.txt file still accessible and were there any changes to it?
- Is your sitemap testing without errors (https://support.google.com/webmasters/answer/183669?hl=en&ref_topic=6080662&rd=2)?
- Did you adjust your Googlebot crawl rate (https://support.google.com/webmasters/answer/48620)?
- Have you had Google's "Fetch as Google" (https://support.google.com/webmasters/answer/6066468) tool re-crawl the URLs?

We recommend exploring Google's Webmaster Tools (https://www.google.com/webmasters/) if you're experiencing issues. Their "Fetch as Google" tool article and their article on troubleshooting sitmemap errors offer specific help for help debugging Googlebot crawl stats in this situation. Google also includes an entire section in their tools documentation on getting additional support (https://support.google.com/webmasters/answer/1249981?hl=en) if you're experiencing trouble.

★ **TIP:** Our debugging articles (/guides/debugging/) contain a variety of troubleshooting tips.

§ Loop detection (/guides/debugging/loop-detection)

Fastly automatically detects loops resulting from service configuration errors. When a loop is detected, Fastly blocks the requests and generates an error message. Loops can occur when the same hostname is configured as both the domain and the origin server, and the CNAME record for the domain is pointed at Fastly. For example, loop detection will be triggered if you set `www.example.com` as the domain and the origin server (/guides/basic-setup/sign-up-and-create-your-first-service#create-your-first-service) in your Fastly service and you add a CNAME DNS record (/guides/basic-setup/adding-cname-records) for `www.example.com` that points at Fastly.

How to avoid triggering loop detection

To avoid triggering loop detection, you should verify the hostname of your origin server is not the same as the domain using one of the following two options:

- Create a DNS hostname (`origin.example.com`) with the appropriate A and AAAA DNS records for your origin server, and use that origin DNS hostname in your Fastly service configuration. This ensures the origin (`origin.example.com`) is different than the domain (`www.example.com`) on your service. We recommend this option. If you make changes to the DNS records for `origin.example.com` in the future, Fastly will automatically detect and use those changes.
- Use an IPv4 address instead of a DNS hostname for your origin's address within your Fastly service's configuration. If the origin server's IP address changes in the future, you'll need to update and activate a new version of your Fastly service configuration.

Example error message

When Fastly detects a loop, an error message similar to the one displayed below will appear in the headers.

```
HTTP/1.1 503 Loop detected
Error-Reason: loop detected
Connection: close
Content-Type: text/plain
Fastly-Host: <hostname>
Fastly-FF: <hostname>
Server: Varnish
```

§ Temporarily disabling caching (/guides/debugging/temporarily-disabling-caching)

Caching can be disabled:

- at the individual URL level,
- at the browser level, and
- at the site level.

Disabling caching at the individual URL level

To disable caching at the individual URL level:

1. Create a request setting that always forces a pass (/guides/basic-configuration/how-request-settings-are-applied).
2. Add a condition to the request setting (/guides/vcl/understanding-the-different-pass-action-behaviors) that looks for specific URLs.
3. Activate the new version of your service to enable the setting.

Disabling caching at the browser level

Theoretically, all browsers should follow the stated rules of the HTTP standard. In practice, however, some browsers don't strictly follow these rules. The following combination of headers seems to force absolutely no caching with every browser we've tested.

```
Cache-Control: no-cache, no-store, private, must-revalidate, max-age=0, max-stale=0, post-check
=0, pre-check=0
Pragma: no-cache
Expires: 0
```

In addition, IE8 has some odd behavior to do with the back button. Adding `Vary: *` to the headers seems to fix the problem.

> ❗ **IMPORTANT:** If you want your content cached in Fastly but not cached on the browser, you must not add these headers on your origin server. Instead, add these as new Headers on the Content page and be sure the Type is set to Response (/guides/basic-configuration/responses-tutorial).

Disabling caching at the site level

You can disable caching at the site level via the Fastly UI or via custom VCL. Via the UI, create a request setting that always forces a pass (/guides/basic-configuration/how-request-settings-are-applied) and then activate the new version of your service to enable the setting. To disable caching at the site level via custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl), add this to your Fastly VCL:

```
sub vcl_recv {
  return(pass);
}
```

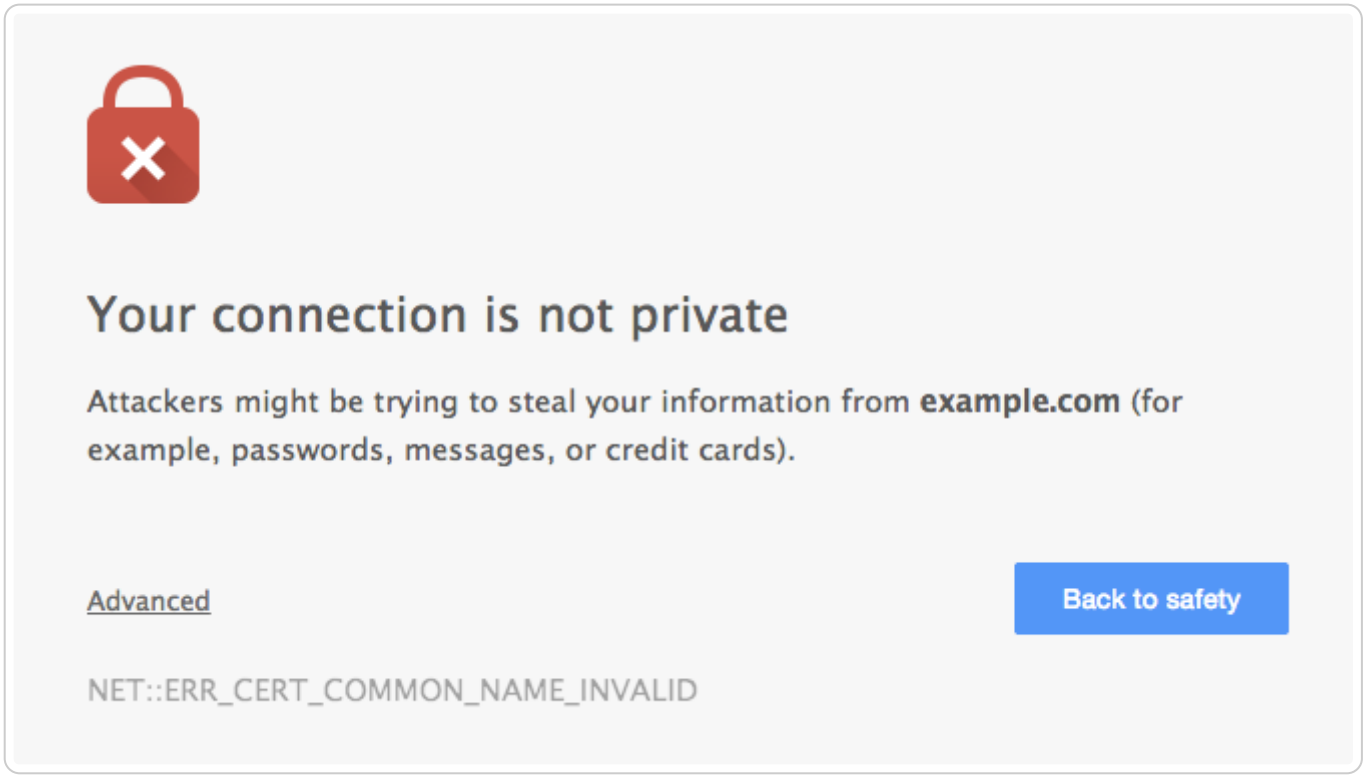
> ❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

---

§ TLS certificate errors (/guides/debugging/tls-certificate-errors)

"Your connection is not private"

If you've recently started testing Fastly services, you may see errors like the following:



These errors appear because your domain has not been provisioned with TLS across the Fastly network. We offer a number TLS options (/guides/detailed-product-descriptions/tls-service-options) that may work for you. Contact support@fastly.com (mailto:support@fastly.com) to begin the provisioning process.

If you don't want to use TLS for your site, set the CNAME DNS record for your domain to point to `global-nossl.fastly.net`. This network endpoint only accepts requests over port 80, and will not expose your users to these certificate errors.

Errors when using Wget

When connecting to a Fastly service using Wget, you may see errors along the lines of

```
ERROR: Certificate verification error for mysite.example.com: unable to get local issuer certif
icate
ERROR: certificate common name `*.a.ssl.fastly.net' doesn't match requested host name `mysite.e
xample.com'.
To connect to mysite.example.com insecurely, use `--no-check-certificate'.
Unable to establish TLS connection.
```

Checking with a browser or cURL will show that there really is no problem, however. The errors appear because a previous version of Wget (wget-1.12-2.fc13) that shipped with some versions of Red Hat Enterprise Linux (RHEL) was buggy and failed to check Subject Alternative Names (SAN) properly.

Upgrading Wget will correct this problem and eliminate the errors. For more information you can read this Red Hat bug report (https://bugzilla.redhat.com/show_bug.cgi?id=674186) or this Debian one (https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=409938). For more information about TLS-related issues, see our TLS guides (/guides/securing-communications/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

§ TLS origin configuration messages (/guides/debugging/tls-origin-configuration-messages)

When you are connecting to origins over TLS (/guides/basic-configuration/connecting-to-origins#setting-the-tls-hostname), you may have errors.

Hostname mismatches

- `Error: Hostname mismatch`

Why the error appears

Your origin server is serving a TLS certificate with a Common Name (CN) or list of Subject Alternate Names (SAN) that does not match the origin host or the origin's SSL hostname setting.

How to fix it

You can fix this by telling Fastly what to match against in the CN or SAN field in your origin's certificate.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **TLS Options** link next to the affected host. The TLS Options page appears.
6. In the **Certificate Hostname** field, type the hostname associated with your TLS certificate. This value is matched against the certificate common name (CN) or a subject alternate name (SAN) depending on the certificate you were issued. For example, if your certificate's CN field is `www.example.com`, type that value for your hostname. If you leave this field blank, the system will use the default host information displayed below this field.

7. Click the **Save** button. Even if you leave the Certificate Hostname field blank to use the default information, you must click the Save button to verify the certificate.
8. Activate a new version of the service to complete the configuration changes.

When using custom VCL (/guides/vcl/uploading-custom-vcl), you can specify the hostname to match against the certificate by using the `.ssl_cert_hostname` field of your origin's definition. For example: `.ssl_cert_hostname = www.example.com;`.

Certificate chain mismatches

- `Error: unable to verify the first certificate`
- `Error: self signed certificate`
- `Error: unable to get local issuer certificate`
- `Error: self signed certificate in certificate chain`
- `Error: unable to get issuer certificate`

Why the errors appear

Your origin server is serving a certificate chain that can not be validated using any of the Certificate Authorities (CAs) that Fastly knows. This can happen for two reasons:

- Your certificate is self-signed or self-issued and you did not provide your generated CA certificate to Fastly for us to use for verification.
- Your certificate is issued by a CA that isn't in Fastly's CA certificates bundle.

How to fix them

In both cases, you can fix your configuration by adding the CA certificate that Fastly should use to verify the certificate to your service configuration:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **TLS Options** link next to the affected host. The TLS Options page appears.
6. In the **TLS CA certificate** field, copy and paste a PEM-formated CA certificate.
7. Click the **Save** button.
8. Activate a new version of the service to complete the configuration changes.

If you are using custom VCL, you can specify the CA for Fastly to use by setting the `.ssl_ca_cert` backend parameter to a PEM encoded CA certificate.

Alternatively, you can get a new certificate issued by a CA in Fastly's CA certificate bundle (e.g., Globalsign).

Connection failures

- `Error: Gethostbyname`
- `Error: Connection timeout`
- `Error: Connection refused`

Why each error appears and how to fix it

For `Gethostbyname` failures, the configured backend Host domain is returning NXDOMAIN. Double check that the DNS settings for your backend are correct.

For `Connection time out` failures, the connection to your server is timing out. Double check that your backend is accessible and responding in a timely fashion.

For `Connection refused` failures, the connection to your server is being refused, potentially by a firewall or network ACL. Double check that you have whitelisted the Fastly IP addresses (/guides/securing-communications/accessing-fastlys-ip-ranges) and that your backend is accessible from our network.

Certificate expirations

`Error: Certificate has expired`

Why the error appears

The certificate your backend server is presenting Fastly has expired and needs to be reissued with an updated validity period.

How to fix it

If this is a self-signed certificate you can perform this update on your own by issuing a new CSR with your private key, creating the corresponding certificate, and installing it on the server.

If this is a CA signed certificate you will need to issue a new CSR with your private key, submit it to your CA, and install the signed certificate they provide you.

SSL and old TLS protocol errors

- `Error: Unknown protocol`

- `Error: SSL handshake failure`
- `Error: TLSv1 alert internal error`

Why the errors appear

Either your origin server is not configured to use TLS or it only supports older, outdated versions of the protocol. We do not support SSLv2 or SSLv3.

How to fix them

If the origin server is configured to use TLS, make sure you are using the latest version of both the server and the TLS library (e.g., OpenSSL). You may have to explicitly enable a newer protocol version. Fastly supports TLSv1, TLSv1.1 and TLSv1.2.

If the origin server is not configured to use TLS, change your service configuration to disable TLS and communicate with it on port 80 instead of port 443:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **TLS Options** link next to the affected host. The TLS Options window appears.
6. From the **Connect to backend using TLS** menu, select **No**.
7. Click the **Save** button.
8. Activate a new version of the service to complete the configuration changes.

RC4 cipher error

- `Error: Using RC4 Cipher`

Why the error appears

When Fastly connects to your origin server using TLS, the only cipher suite your server supports for establishing a connection is the RC4 cipher. This cipher is considered to be unsafe for general use and should be deprecated.

How to fix it

You can fix this on your origin by using the latest version of both the server and the TLS library (e.g., OpenSSL) and ensuring the cipher suites offered are tuned to best practices. You may need to explicitly blacklist the RC4 cipher.

§ Using GET instead of HEAD for command line caching tests (/guides/debugging/using-get-instead-of-head-for-command-line-caching-tests)

If you're testing on the command line to determine an object's caching status, then use GET instead of HEAD. For example:

```
curl -svo /dev/null www.example.com
```

Default caching behavior of HTTP verbs

By default, the results of GET requests are cached. HEAD requests are not proxied as is, but are handled locally if an object is in cache or a GET is done to the backend to get the object into the cache. Anything other than HEAD or GET requests are proxied and not cached by default. However, POSTs *can* be cached if required, and caching of that can be keyed off of the contents of the POST request body if that is below 2K. We've posted an example of how to do this (/guides/vcl/manipulating-the-cache-key#using-a-post-request-body-as-a-cache-key).

• Guides (/guides/) > Diagnostics and performance > Performance tuning (/guides/performance-tuning/)

§ Changing origins based on user location (/guides/performance-tuning/changing-origins-based-on-user-location)

Fastly allows you to change origin servers based on the user's geographic location. This is useful when you need to serve different content to users who are in different locations. For example, you could change origin servers to serve a restricted version of your website to users in a different country.

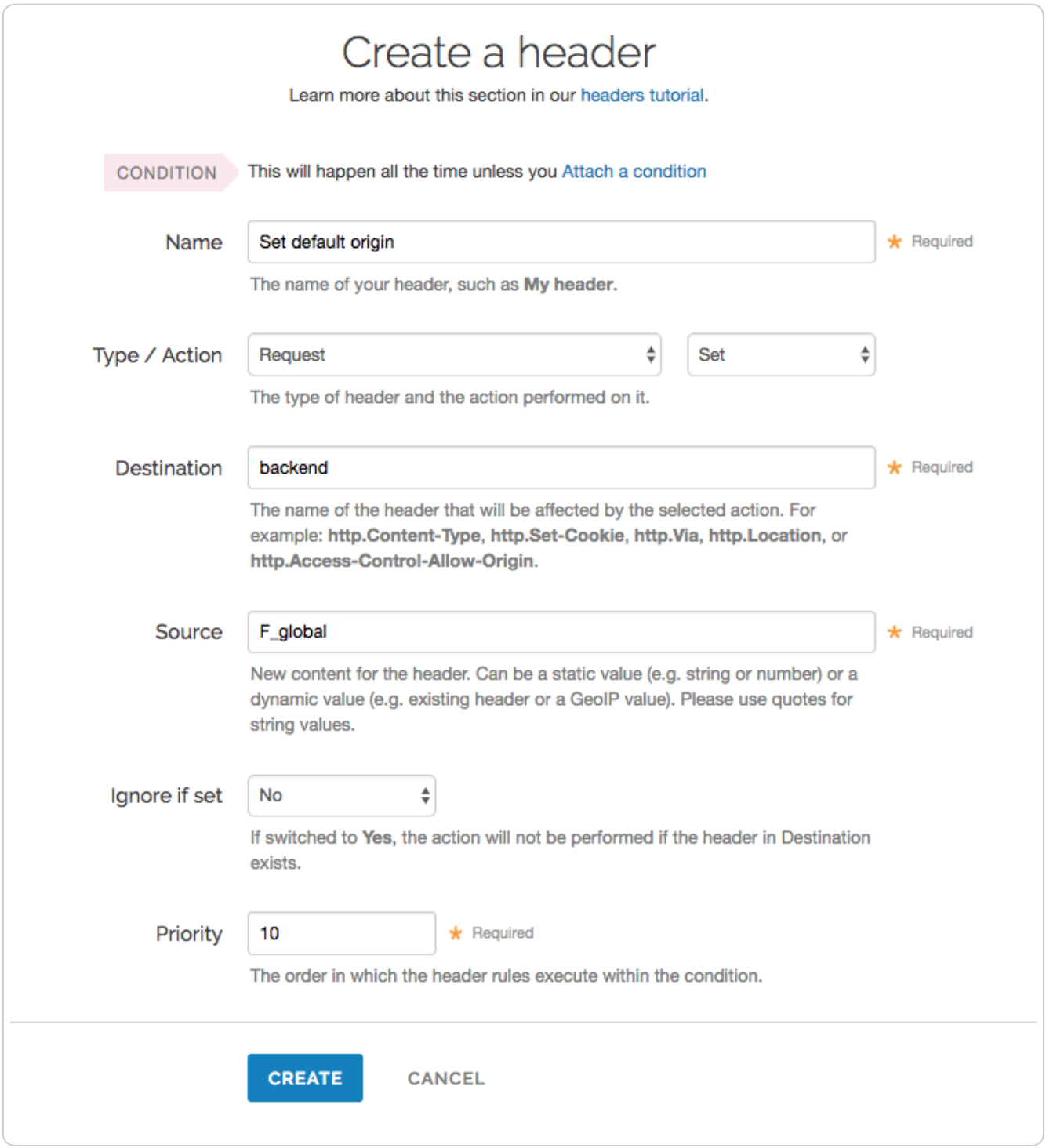
Using the web interface

You can use the web interface to create the headers and the condition.

Creating the header for the default origin server

First, create a header for the default origin server to serve content to the majority of users. Follow these instructions to create the header:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `Set default origin`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `backend`.
   - In the **Source** field, type the name of origin server you want to serve content to the majority of users (here it's `F_global`). Preview the VCL (/guides/vcl/previewing-and-testing-vcl-before-activating-it) to find the name of the origin server.

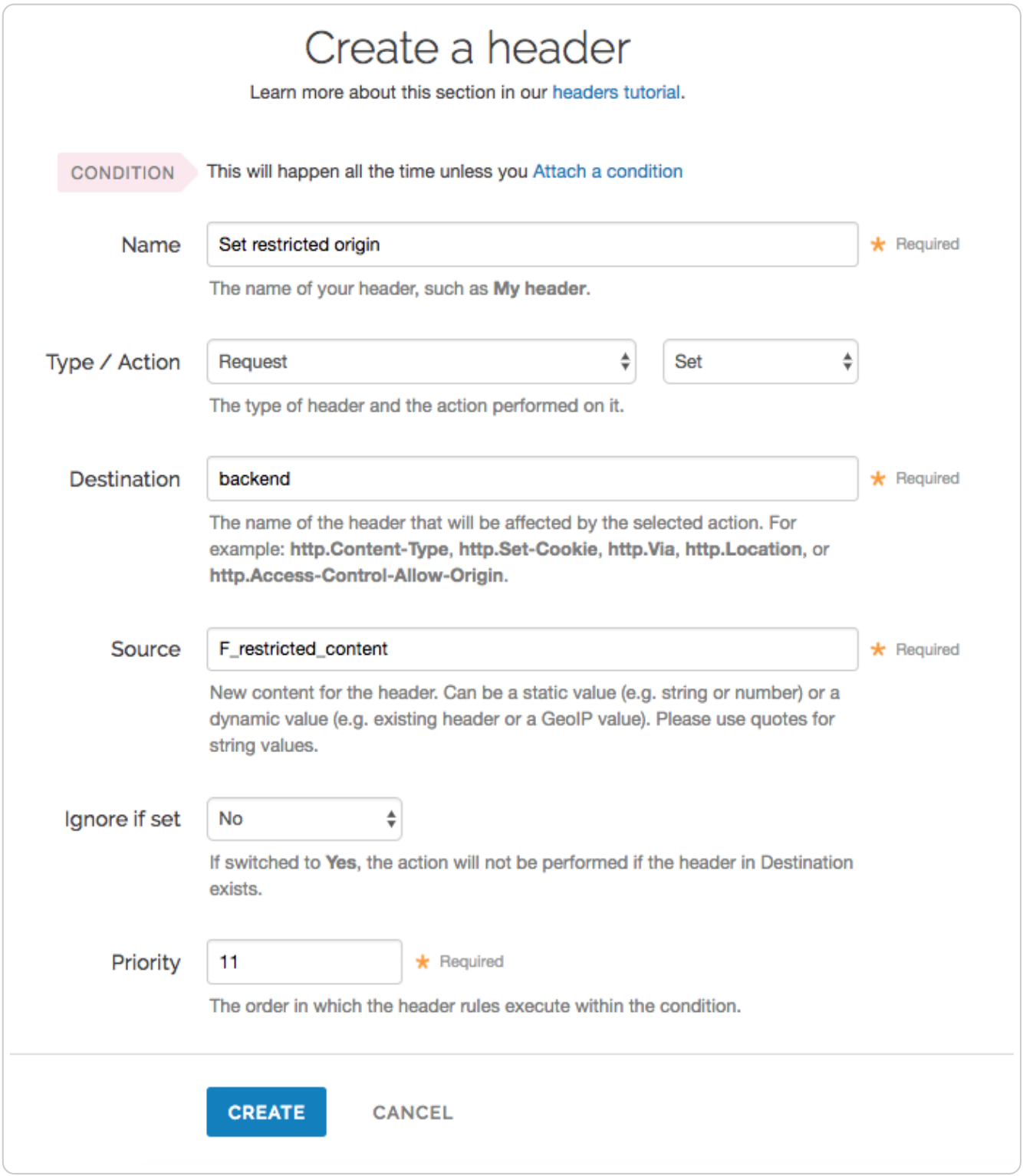
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button.

Creating the header for the restricted origin server

Now, create a header for the restricted origin server to serve content to the users residing in the countries specified in the condition. Follow these instructions to create the header:

1. Click the **Content** link. The Content page appears.
2. Click the **Create header** button. The Create a header window appears.



3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `Set restricted origin`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `backend`.

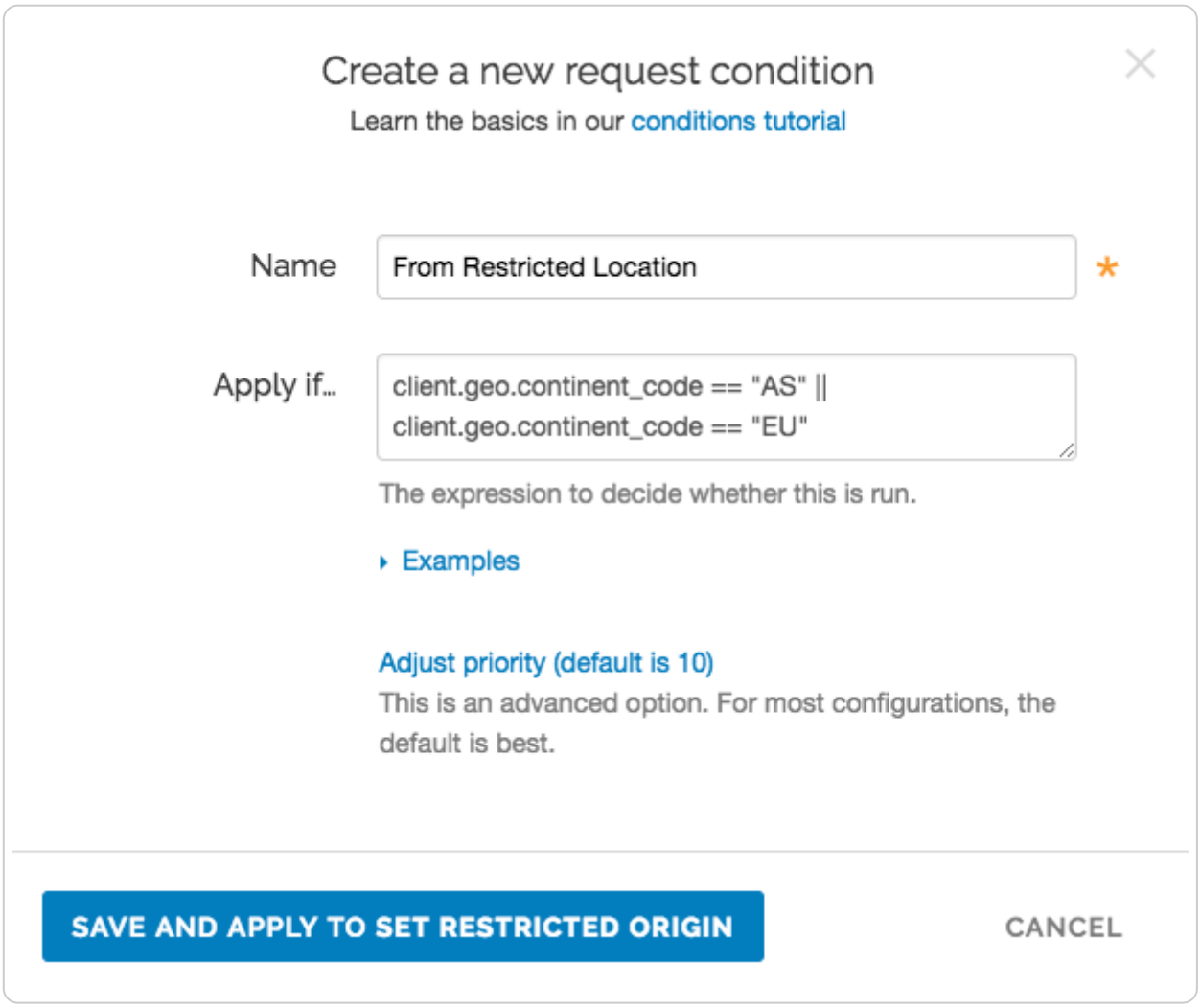
- In the **Source** field, type the name of restricted origin server you want to serve content to the users residing in the countries specified in the condition (here it's `F_restricted_content`). Preview the VCL (/guides/vcl/previewing-and-testing-vcl-before-activating-it) to find the name of the origin server.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `11`.

4. Click the **Create** button.

Creating a condition for the restricted origin header

Finally, create a condition for the restricted origin header. The condition checks the geolocation header (/guides/vcl/geolocation-related-vcl-features). If the user's geolocation matches a location specified in the condition, Fastly uses the restricted origin server. Follow these instructions to create the condition:

1. Click the **Content** link. The Content page appears.
2. In the Headers section, click the **Attach a condition** link next to the **Set restricted origin** header. The Create a new request condition window appears.



3. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `From Restricted Location`).
   - In the **Apply if** field, type a request condition. For example, to send all users in Asia and Europe to the restricted origin server, type `client.geo.continent_code == "AS" || client.geo.continent_code == "EU"`. See Geolocation-related VCL features (/guides/vcl/geolocation-related-vcl-features) for more information.
4. Click the **Save and apply to** button.
5. Click the **Activate** button to deploy your configuration changes.

Using custom VCL

If you'd prefer not to use the web interface, you can use custom VCL to configure your service to change origin servers based on the user's geographic location. Use the following VCL as a starting point:

```
  # default conditions
  set req.backend = F_global;

  # Use restricted content if the user is in Asia, France or Germany
  if (client.geo.continent_code == "AS" || client.geo.country_code == "FR" || client.geo.countr
y_code == "DE") {
    set req.backend = F_restricted_content;
  }
```

IMPORTANT: The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

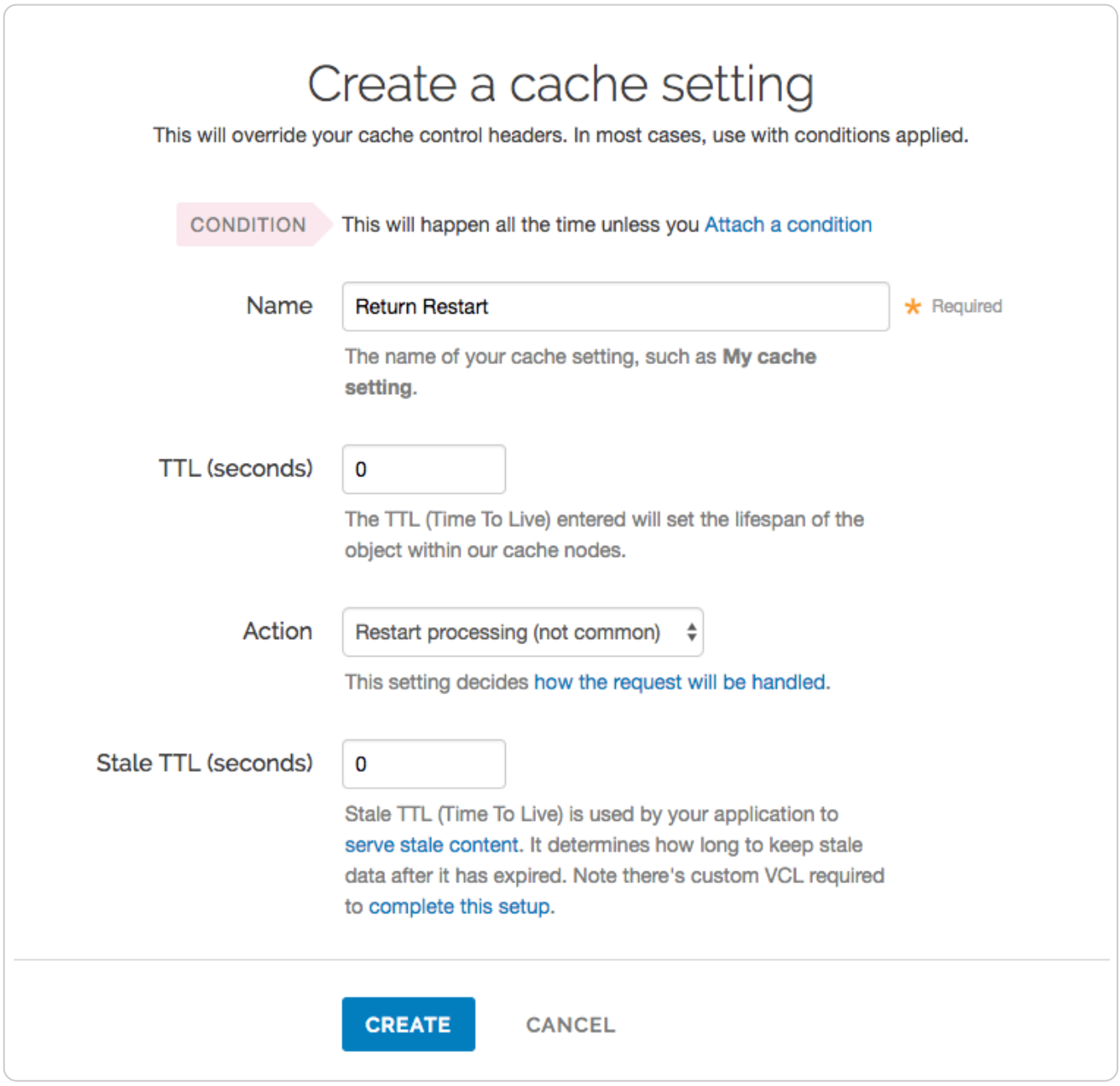
§ Checking multiple backends for a single request (/guides/performance-tuning/checking-multiple-backends-for-a-single-request)

Using a restart is a good option to check multiple backends for a single request. This can be created using a cache setting rule (/guides/tutorials/cache-control-tutorial) and request headers.

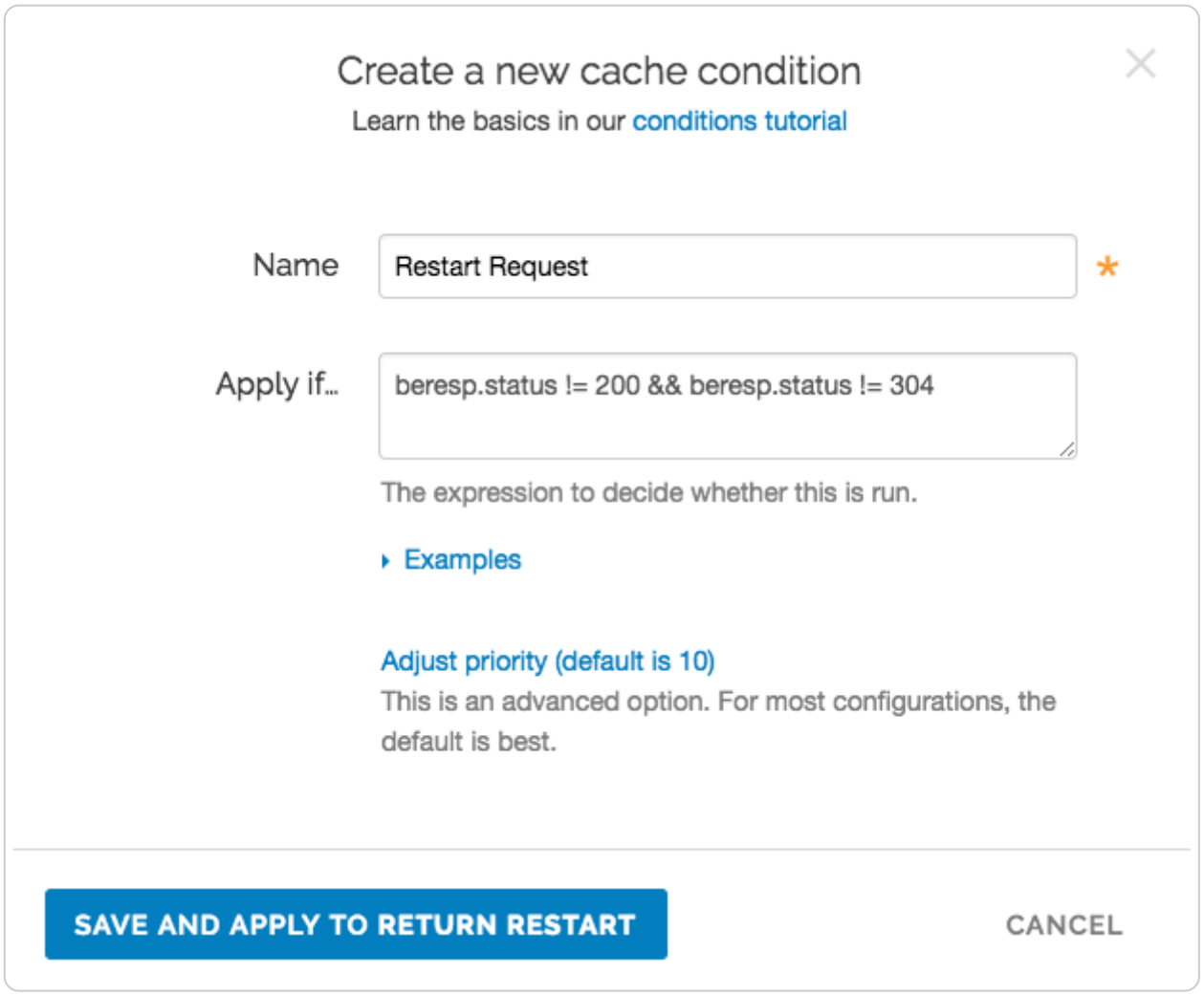
Create a new cache setting rule

Follow these steps to create a cache restart within vcl_fetch.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Create cache setting** button. The Create a cache setting page appears.



6. Fill out the **Create a cache setting** fields as follows:
   - In the **Name** field, type `Return Restart` (or any meaningful, preferred name).
   - In the **TTL (seconds)** field, type `0`.
   - From the **Action** menu, select **Restart processing**.
   - In the **Stale TTL (seconds)** field, type `0`.
7. Click the **Create** button. The new cache setting appears on the Settings page.
8. On the **Settings** page, click the **Attach a condition** link next to the cache setting you just created. The Create a new cache condition window appears.

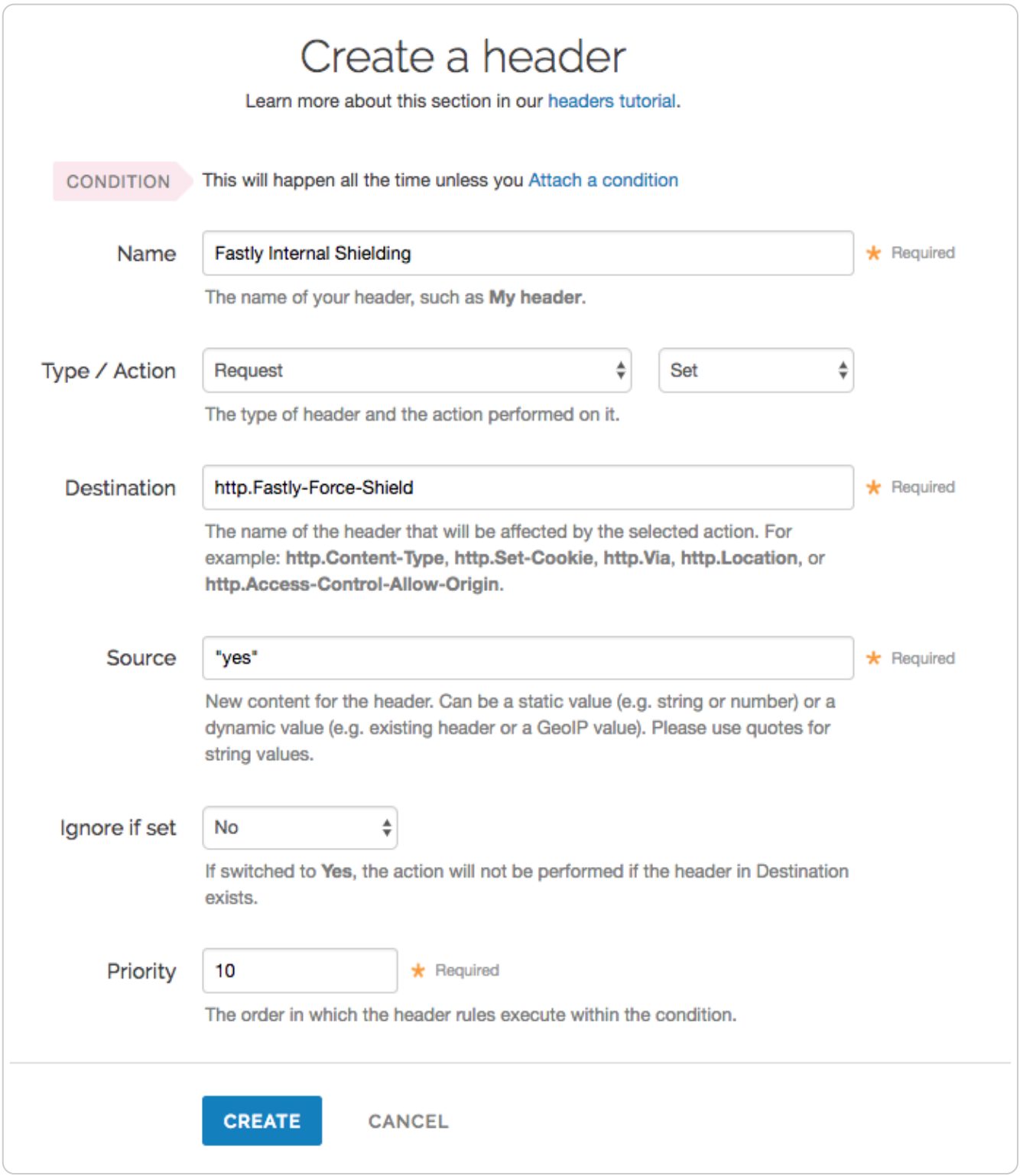


9. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, type `Restart Request` (or any meaningful, preferred name).
   - In the **Apply if** field, type `beresp.status != 200 && beresp.status != 304`.
10. Click the **Save and apply to** button to create the condition.

Create new request headers

Follow these steps to create a request header within vcl_recv.

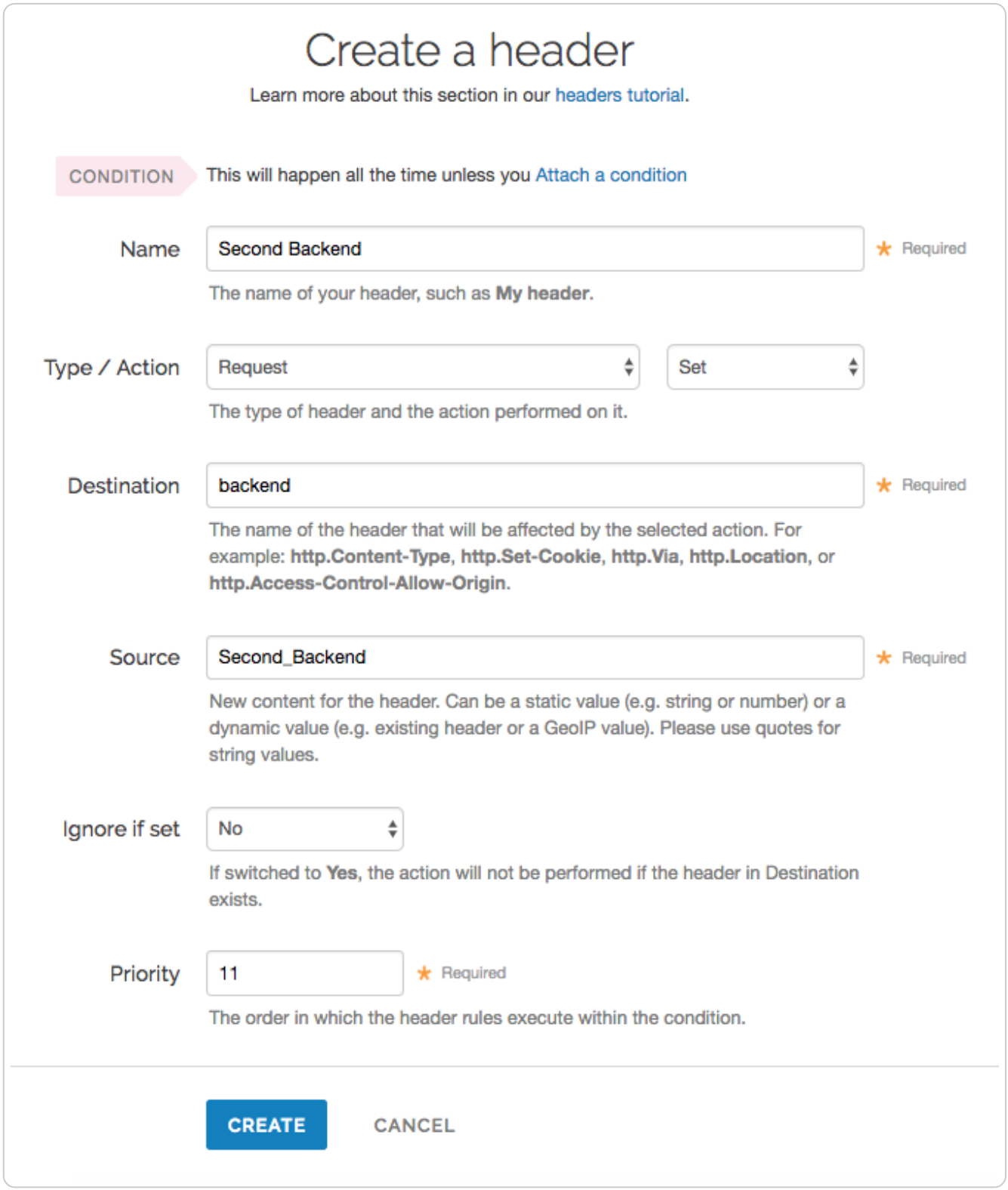
1. Click the **Content** link. The Content page appears.
2. Click the **Create header** button. The Create a header page appears.



3. Fill out the **Create a new header** fields as follows:
   - In the **Name** field, type `Fastly Internal Shielding` (or any meaningful, preferred name).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Fastly-Force-Shield`.
   - In the **Source** field, type `"yes"`.
   - From the **Ignore if set** menu, select **No**.

- In the **Priority** field, type `10`.

4. Click the **Create** button. The new header appears on the Content page.
5. Click the **Create header** button to create another header to switch to the next backend. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:

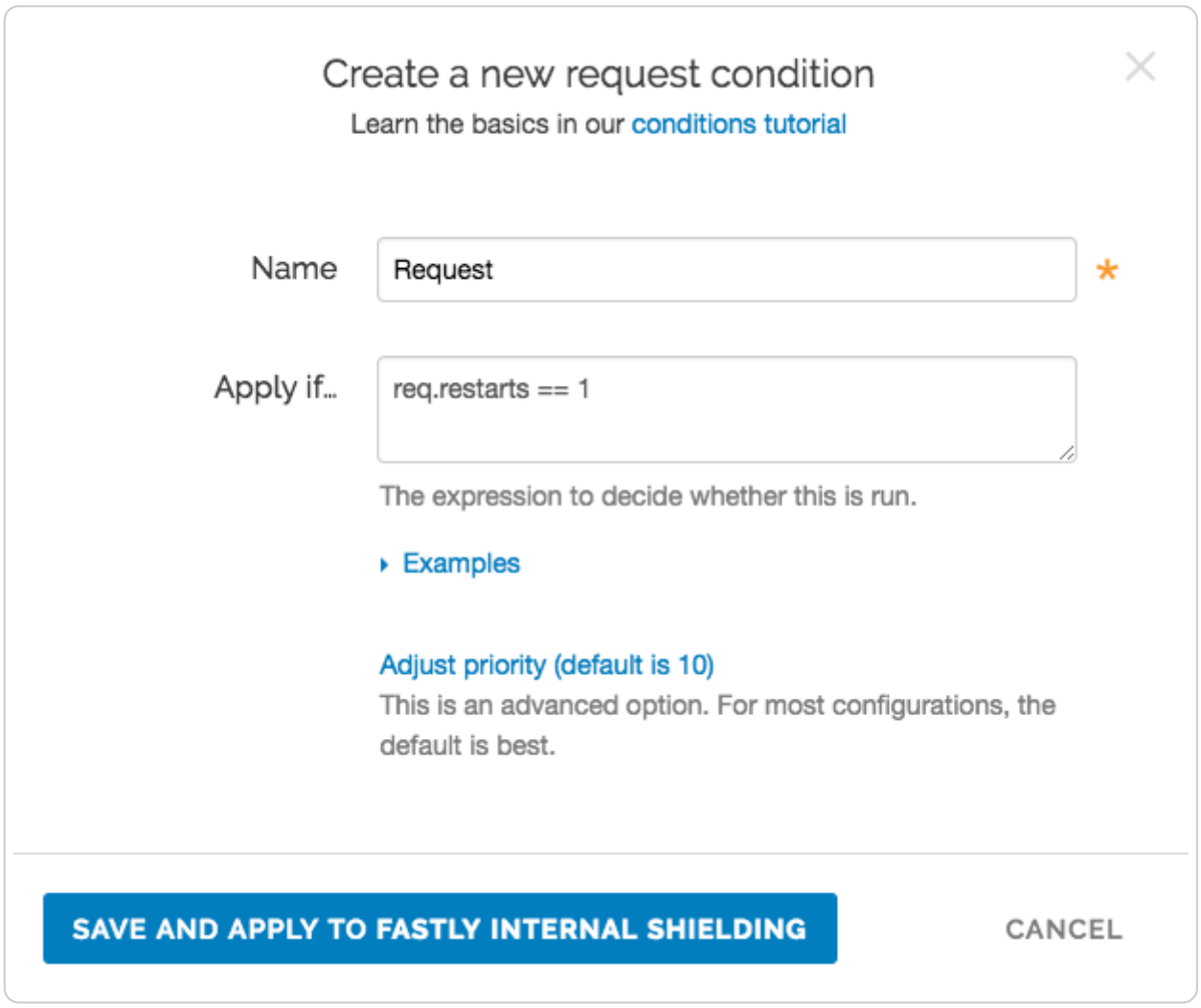
- In the **Name** field, type `Second Backend` (or any meaningful, preferred name).
- From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
- In the **Destination** field, type `backend`.
- In the **Source** field, type `Second_Backend` (this should match the name of your other backend).
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `11`.

7. Click the **Create** button. The new header appears on the Content page.

Create new header conditions

Follow these steps to create conditions for the headers.

1. On the **Content** page, click the **Attach a condition** link next to one of the headers you just created. The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:

- In the **Name** field, type `Req.request` (or any meaningful, preferred name).
- In the **Apply if** field, type `req.restarts == 1`.

3. Click **Save and apply to**. The condition appears on the Content page.
4. Repeat steps 1-3 for the other header.
5. Click the **Activate** button to deploy your configuration changes.

---

§ Controlling caching (/guides/performance-tuning/controlling-caching)

How long Fastly caches content

The maximum amount of time we cache content depends on a number of factors including the TTL (Time To Live) and Grace Period, how often an object gets accessed, and how busy other customers are. Setting TTL and Grace Period to a week, possibly even two weeks should be absolutely fine. For more information about controlling how long Fastly caches your resources, start with our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

In general, we will honor any cache-control headers (/guides/basic-concepts/how-fastlys-cdn-service-works) you send to us from your origin; however, if you do not send any cache-control headers, you will most likely reach the Default TTL limit for your service. You can change this limit on the Configuration page (/guides/basic-concepts/about-the-web-interface-controls#about-the-configure-page).

Changing caching times for different users

You can change the caching times for different users through Surrogate-Control headers defined by the W3C (http://www.w3.org/TR/edge-arch/). If, for example, you wanted Fastly to cache something for a month (clearing with API purges, if necessary) but you also wanted to set a maximum age of a single day for users viewing that object in a browser, then you could return the HTTP header:

```
Surrogate-Control: max-age=2629744
Cache-Control: max-age=86400
```

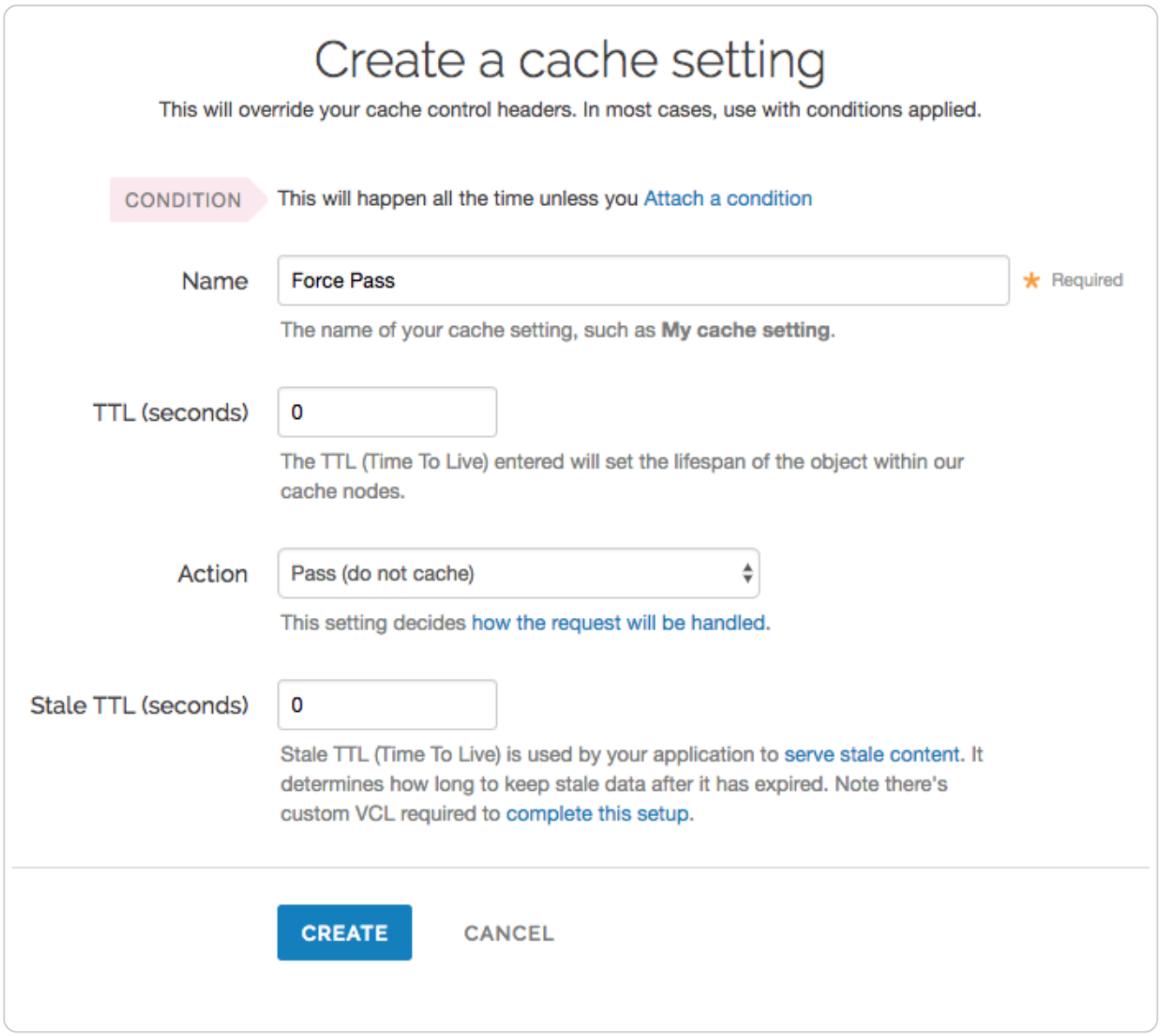
The Surrogate Control header in this example tells Fastly to cache the object for a maximum of 2629744 seconds (one month). The Cache Control header in this example tells the browser to cache the object for a maximum of 86400 seconds (1 day).

For more information about controlling caching, see our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

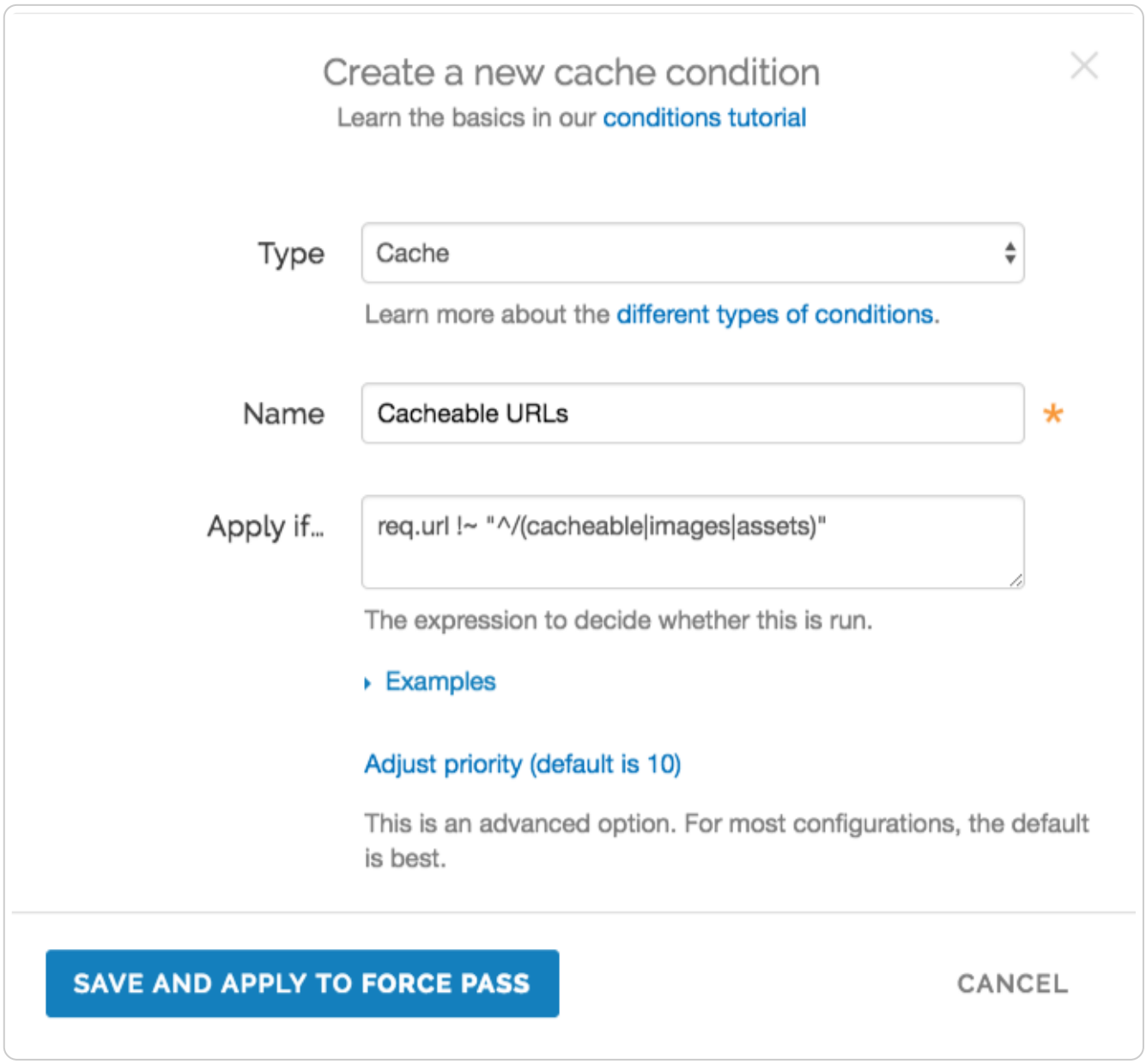
Conditionally preventing pages from caching

To conditionally prevent pages from caching, follow the steps below.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Create cache setting** button to create a new cache setting. The Create a cache setting page appears.



6. Create a new cache setting and then click the **Create** button. The new cache setting you created appears on the Settings page.
7. Click the **Attach a condition** link to the right of the newly created cache setting. The Create a new cache condition window appears.



8. Create a condition that matches the URLs you want and then click the **Save and apply to** button. In this example, we set the condition to look for URLs containing `/cacheable`, `/images`, or `/assets`. If the condition finds them, the URLs should be cached. If the condition doesn't find them, the URLs are explicitly not cached by the apply if statement shown above.
9. Click the **Activate** button to deploy your configuration changes.

★ **TIP:** You can use these steps to override default caching based on a backend response (/guides/basic-configuration/overriding-caching-defaults-based-on-backend-responses).

Caching action descriptions

You can use actions to tell Fastly what to do with cached objects and what to do with additional cache configurations as a result. The following actions are available:

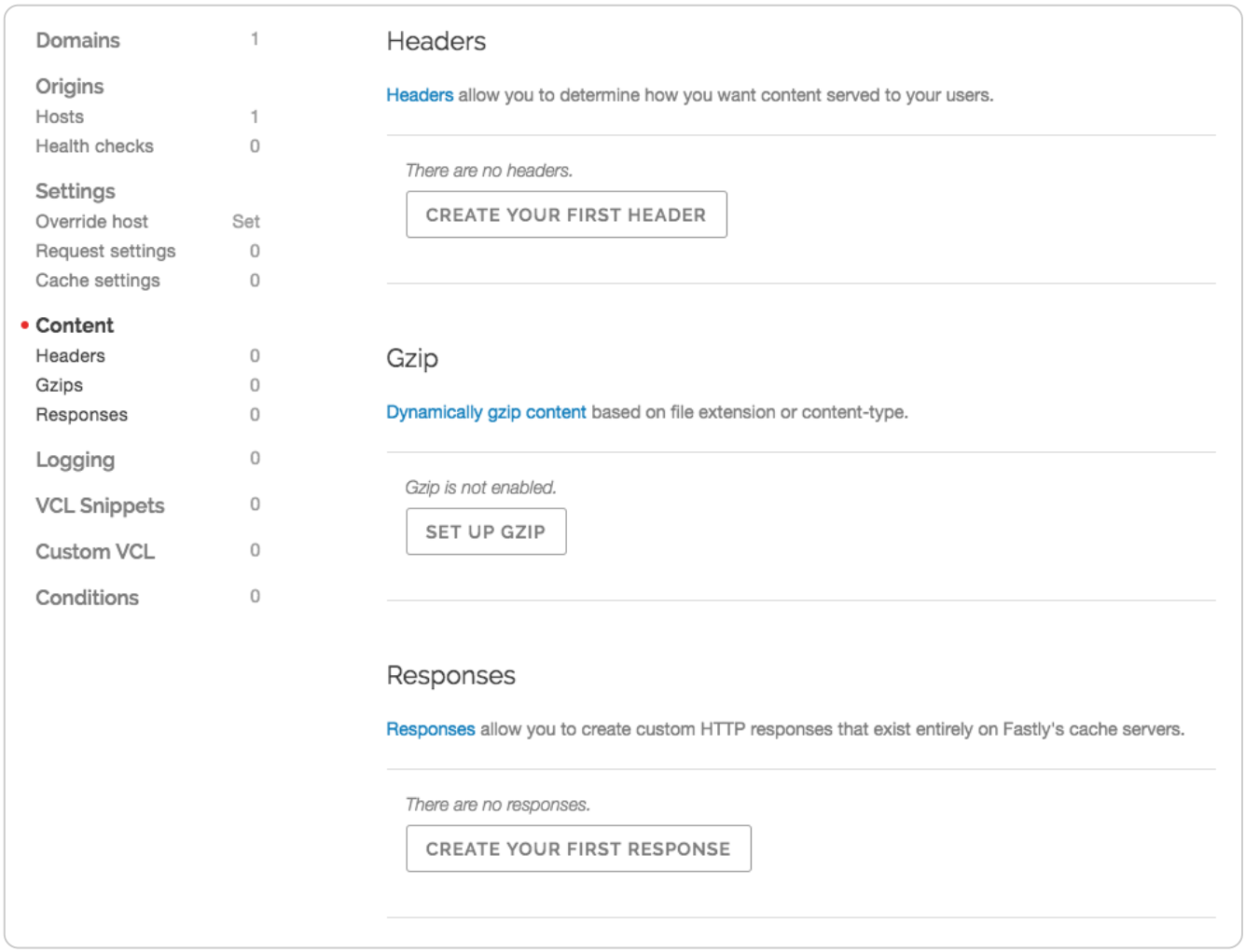
- **Do nothing now** - Only set the TTL or stale TTL.
- **Deliver** - Deliver the object to the client. Usually returned from `vcl_fetch`.
- **Pass** - Pass the request and subsequent response to and from the origin server without caching the object. Usually returned from `vcl_recv`.
- **Restart** - Restart the request processing for the object. You can restart the processing of the whole transaction. Changes to the `req` object are retained.

---

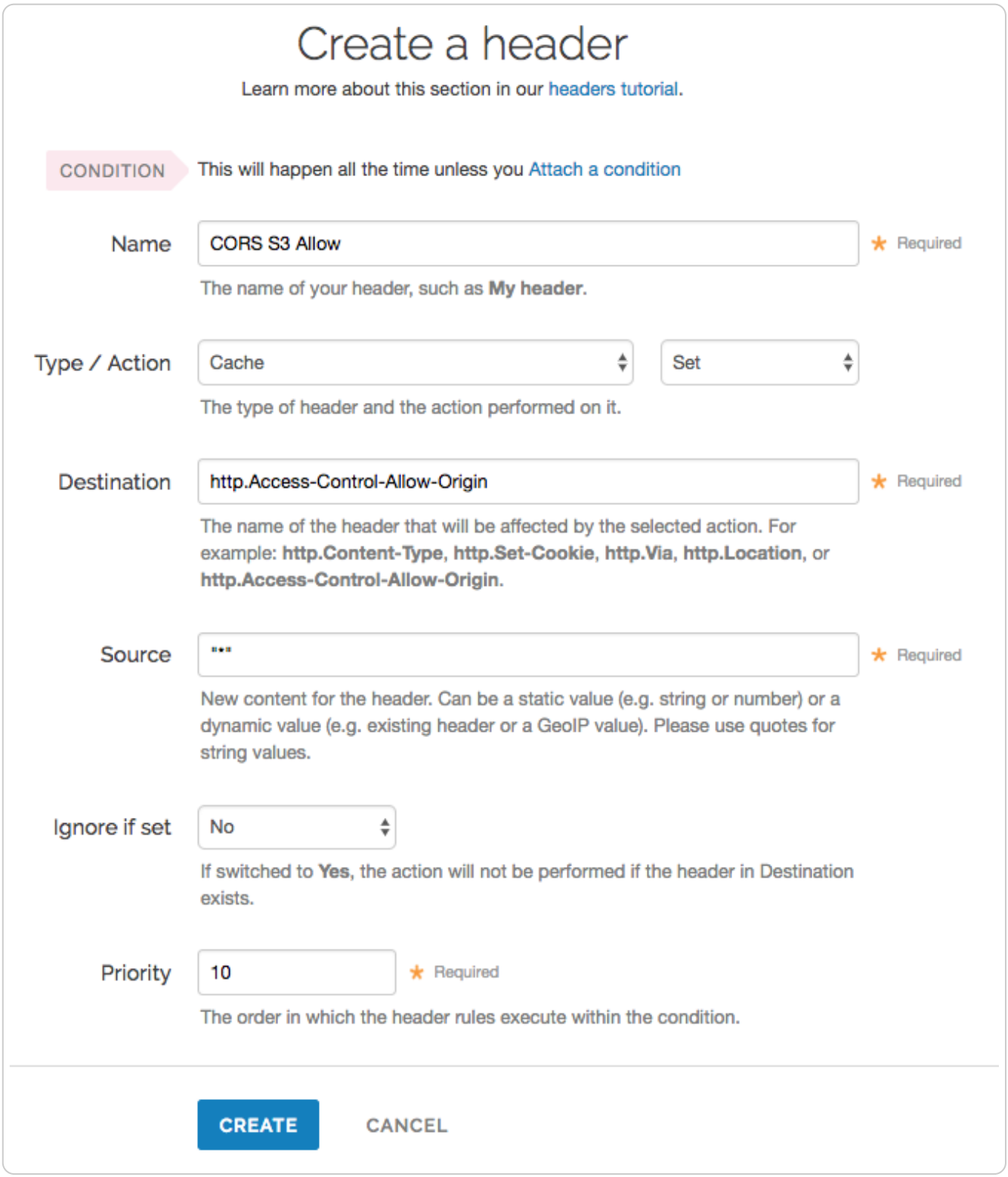
§ Enabling cross-origin resource sharing (CORS) (/guides/performance-tuning/enabling-cross-origin-resource-sharing)

We recommend enabling CORS (Cross-Origin Resource Sharing (http://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html)) when using Amazon S3 (/guides/integrations/amazon-s3) as your backend server. To enable CORS, set up a custom HTTP header for your service by following the steps below.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the new header (e.g., `CORS S3 Allow`). This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Access-Control-Allow-Origin`.
   - In the **Source** field, type `"*"`.

- Leave the **Ignore if set** menu and the **Priority** field set to their default values.

7. Click the **Create** button. The new header appears on the Content page.
8. Click the **Activate** button to deploy your configuration changes.

❗ **IMPORTANT:** Objects already cached won't have this header applied until you purge them (/guides/purging/).

Test it out

Running the command `curl -I example.tld/path/to/resource` should include similar information to the following in your header:

```
Access-Control-Allow-Origin: http://example.tld
Access-Control-Allow-Methods: GET
Access-Control-Expose-Headers: Content-Length, Connection, Date...
```

---

§ Enabling global POPs (/guides/performance-tuning/enabling-global-pops)

The sun never sets on the Fastly empire, but how can you take full advantage? Simply set your CNAME record (/guides/basic-setup/adding-cname-records) to `nonssl.global.fastly.net` for non TLS traffic. You'll now have access to all of our worldwide POPs (https://www.fastly.com/network-map) as they come online. We don't restrict POP access. Instead, you control it.

How to check if your CNAME is set to nonssl.global.fastly.net

Run the following command in your terminal:

```
$ dig www.example.com +short
```

Your output should appear similar to the following:

```
nonssl.global.fastly.net.
151.101.117.57
```

If you don't see `nonssl.global.fastly.net.` in your output, then your CNAME isn't properly set. We link to instructions for setting your CNAME (/guides/basic-setup/adding-cname-records) for a number of popular providers.

Instead of using the above command in your terminal, you can also use various online DNS checking tools, such as the OpenDNS Cache Check (https://cachecheck.opendns.com/).

Limiting POP use to North America and the European Union

You can route your traffic through Fastly's North American and European Union POPs only. If you're not using TLS, simply set your CNAME record (/guides/basic-setup/adding-cname-records) to `nossl.us-eu.fastly.net.` instead of `nonssl.global.fastly.net.`. If you're using TLS, see our guide on CNAME records (/guides/basic-setup/adding-cname-records) to find the appropriate entry.

---

§ Failover configuration (/guides/performance-tuning/failover-configuration)

This guide describes how to configure a failover origin server. A failover (backup) server ensures you can maintain availability of your content if your primary server is not available.

Before you begin

Before you configure failover, keep in mind the following:

- To configure a failover origin server you must make sure you have health checks (/guides/basic-configuration/health-checks-tutorial) configured for your primary server. If you configure your failover server but don't configure health checks (/guides/basic-configuration/health-checks-tutorial) on the primary server, the failover won't work properly if your primary server stops responding.
- Many customers configure load balancing at the same time they configure failover functionality. Our guide on configuring load balancing (/guides/performance-tuning/load-balancing-configuration) can show you how.

Configuring a failover origin server

Once you've confirmed health checks are configured, you must:

1. Turn off automatic load balancing on both the primary origin server and the server that will become your failover.
2. Create headers that configure both the primary and failover origin servers.
3. Create a header condition that specifies exactly when to use the failover server.

Turn off automatic load balancing

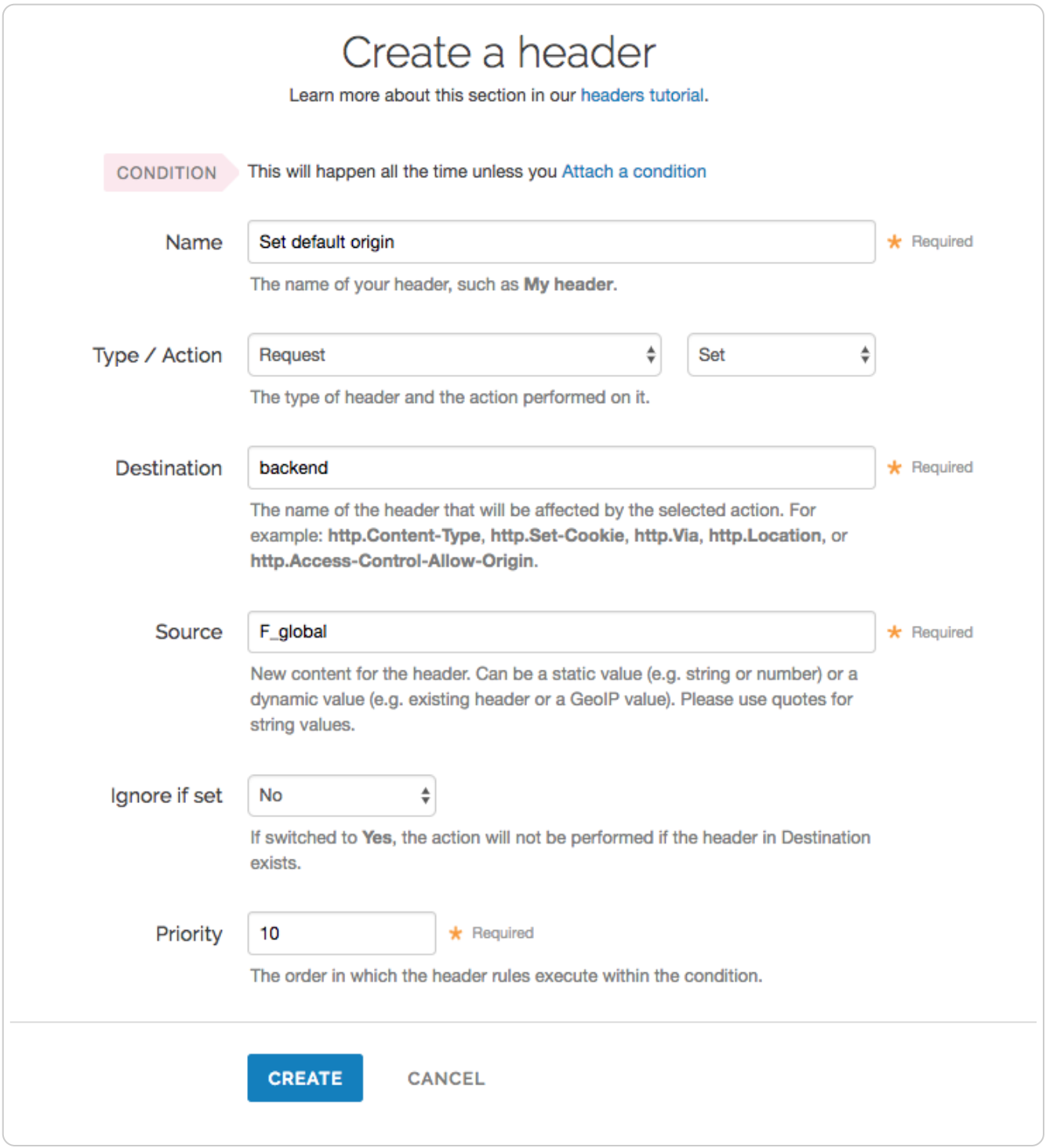
To configure a failover origin server you must turn off automatic load balancing for both the server that will act as your primary origin server and the server that will become your failover origin server.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the name of the origin server you want to configure. The Edit this host page appears.
6. From the **Auto load balance** menu, select **No**.
7. Click the **Update** button to apply the changes.

Configure the primary and failover origin servers

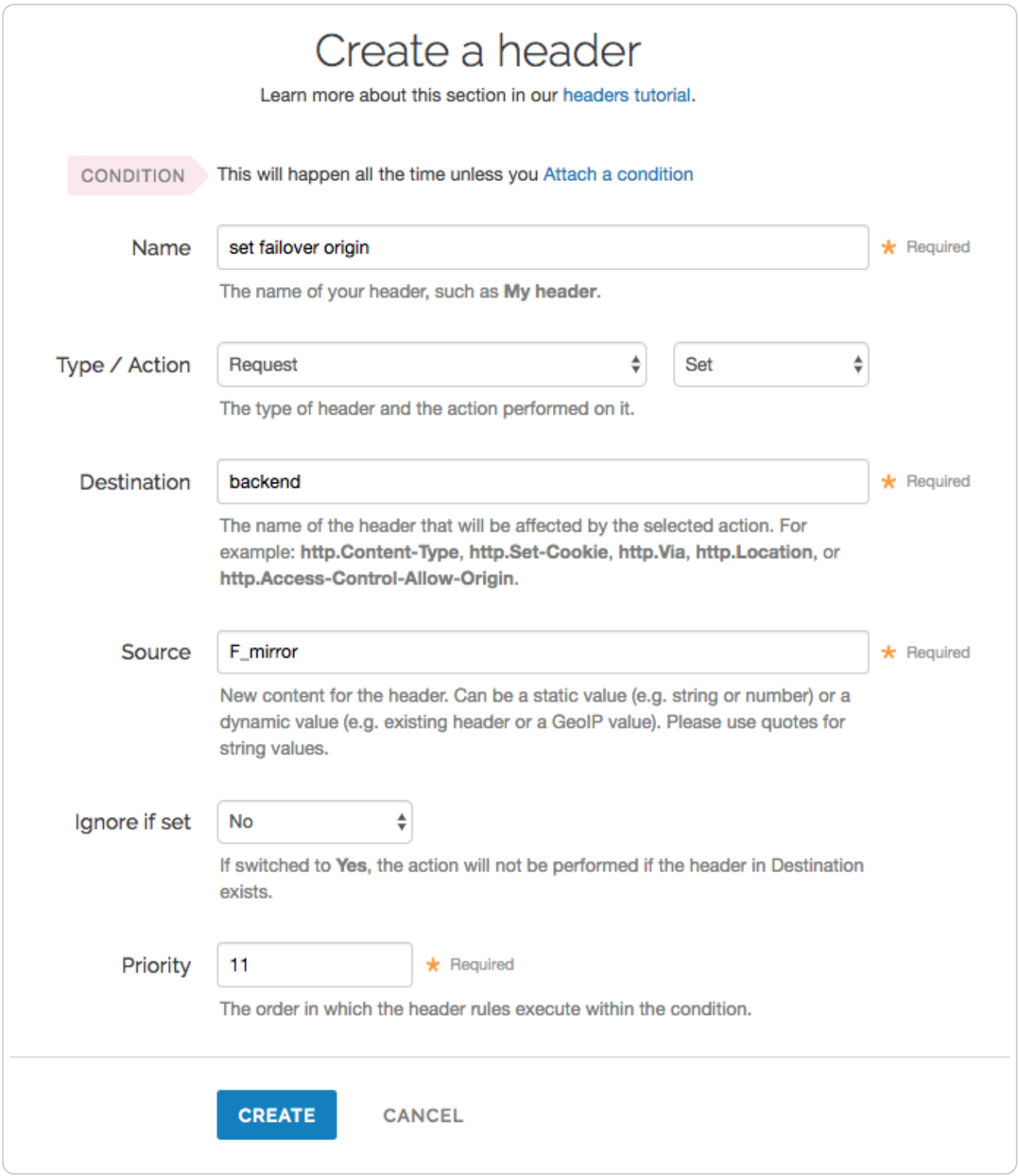
Once you've turned off automatic load balancing, create two new request headers, one each for your primary and failover servers.

1. Click the **Content** link. The Content page appears.
2. Click the **Create header** button to create the first request header. The Create a header window appears.



3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the header. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header that will be affected by the selected action.
   - In the **Source** field, type where the new content for the header comes from.
   - Leave the **Ignore if set** and **Priority** controls at their default settings.

4. Click the **Create** button to create the first header. A new header appears on the Content page.
5. Click the **Create header** button to create a second request header. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the header. This name is displayed in the Fastly web interface.

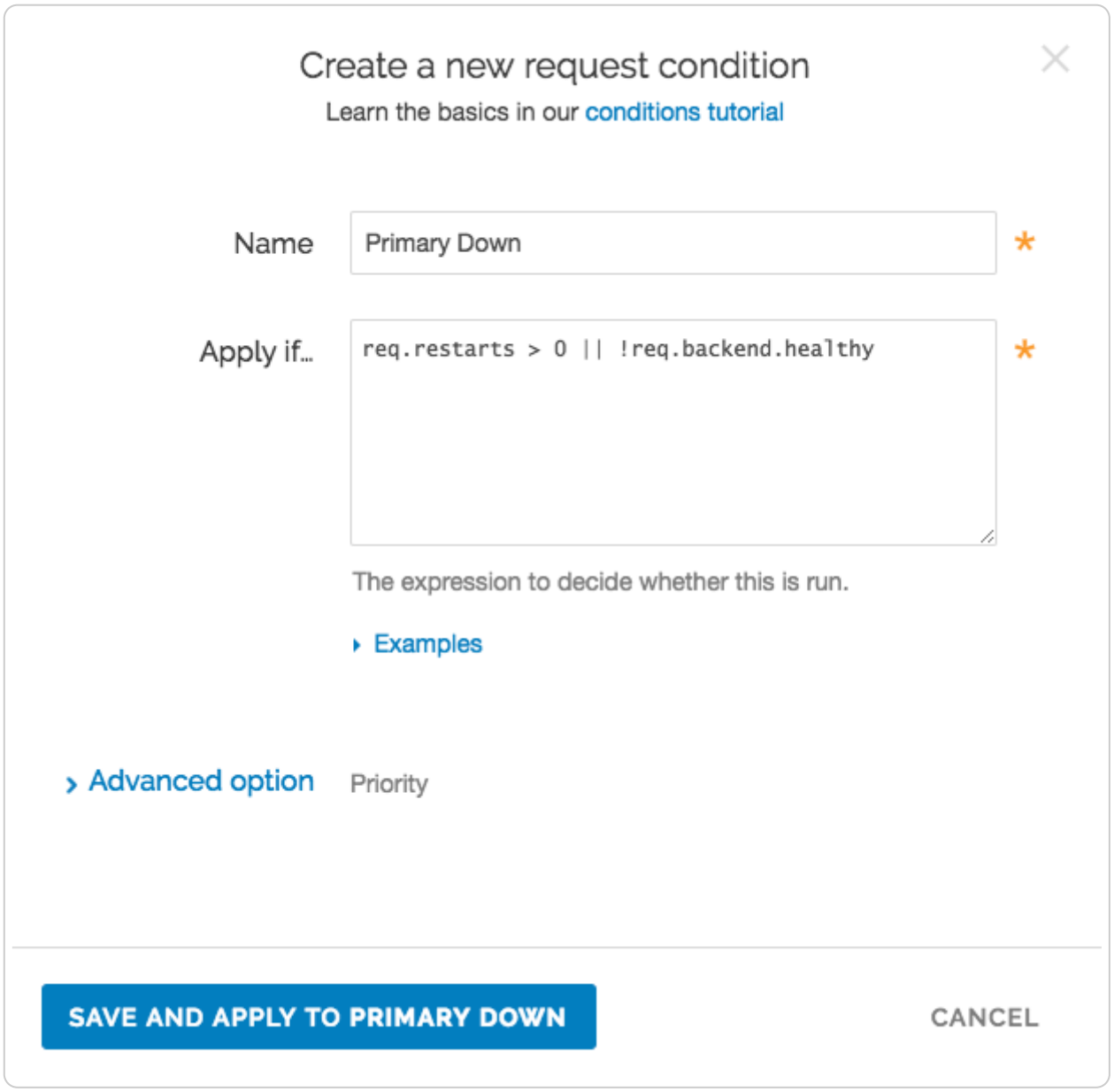
- From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
- In the **Destination** field, type the name of the header that will be affected by the selected action.
- In the **Source** field, type where the new content for the header comes from.
- Leave the **Ignore if set** control at the default setting.
- In the **Priority** field, type a number at least one higher than the priority you set on the primary server's request header. For example, if you left the first header's priority set to the default, `10`, you would set the second header's priority to `11` or higher.

7. Click the **Create** button to create the second header. A new header appears on the Content page.

Specify when to use the failover server

Once you've configured your primary and failover servers, create an associated header condition that specifies exactly when the failover server should be used.

1. On the **Content** page, click the **Attach a condition** link next to the new header you just created for the failover origin server. The Create a new request condition window appears.



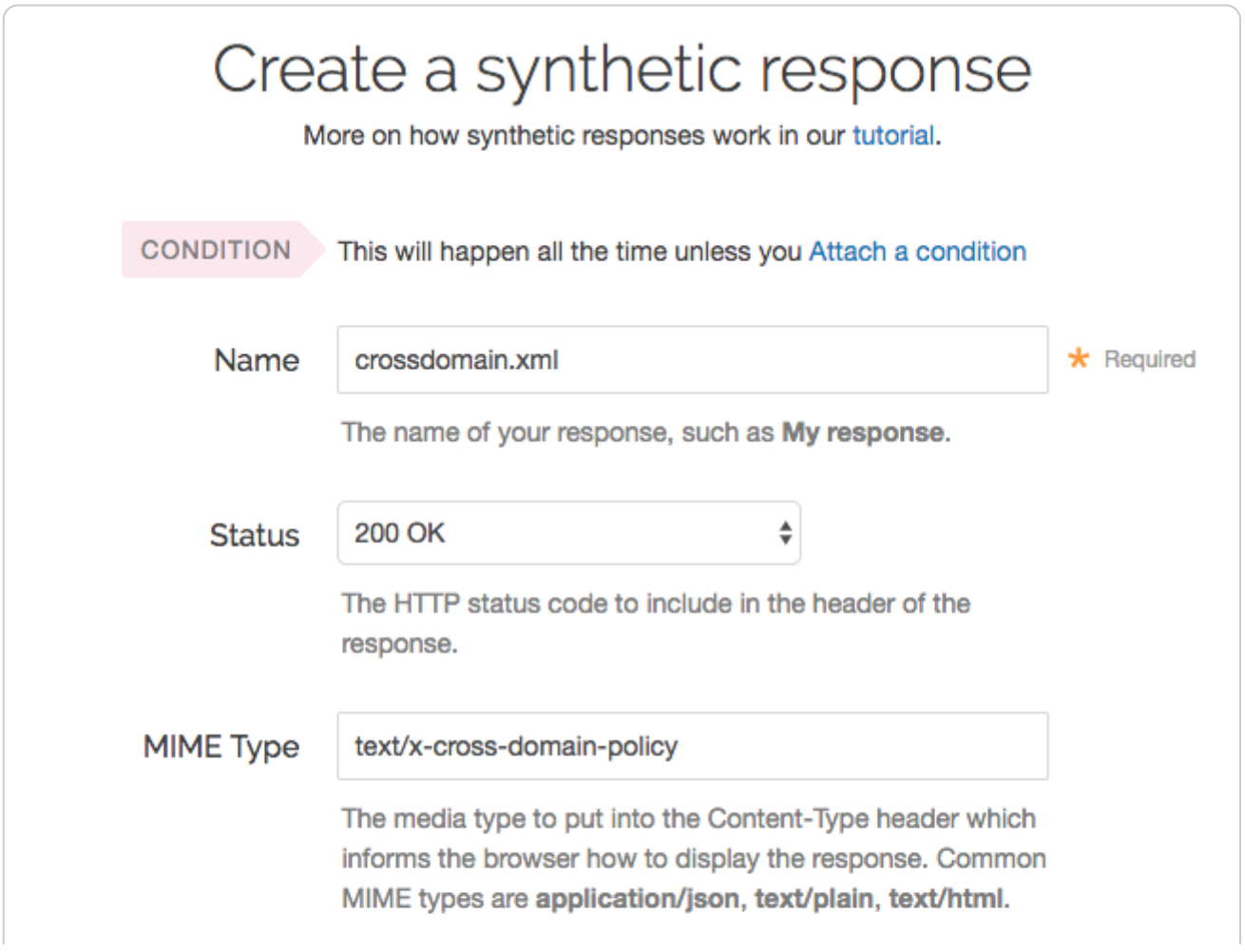
2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Primary Down`).
   - In the **Apply if** field, type the appropriate request condition that will be applied. For example, `req.restarts > 0 || !req.backend.healthy` would tell the system only to use the failover server if the number of restarts is more than 0 or the origin is unhealthy.
3. Click the **Advanced Options** link.
4. In the **Priority** field, type `11`.
5. Click the **Save and apply to** button to create the new condition for the header.
6. Click the **Activate** button to deploy your configuration changes.

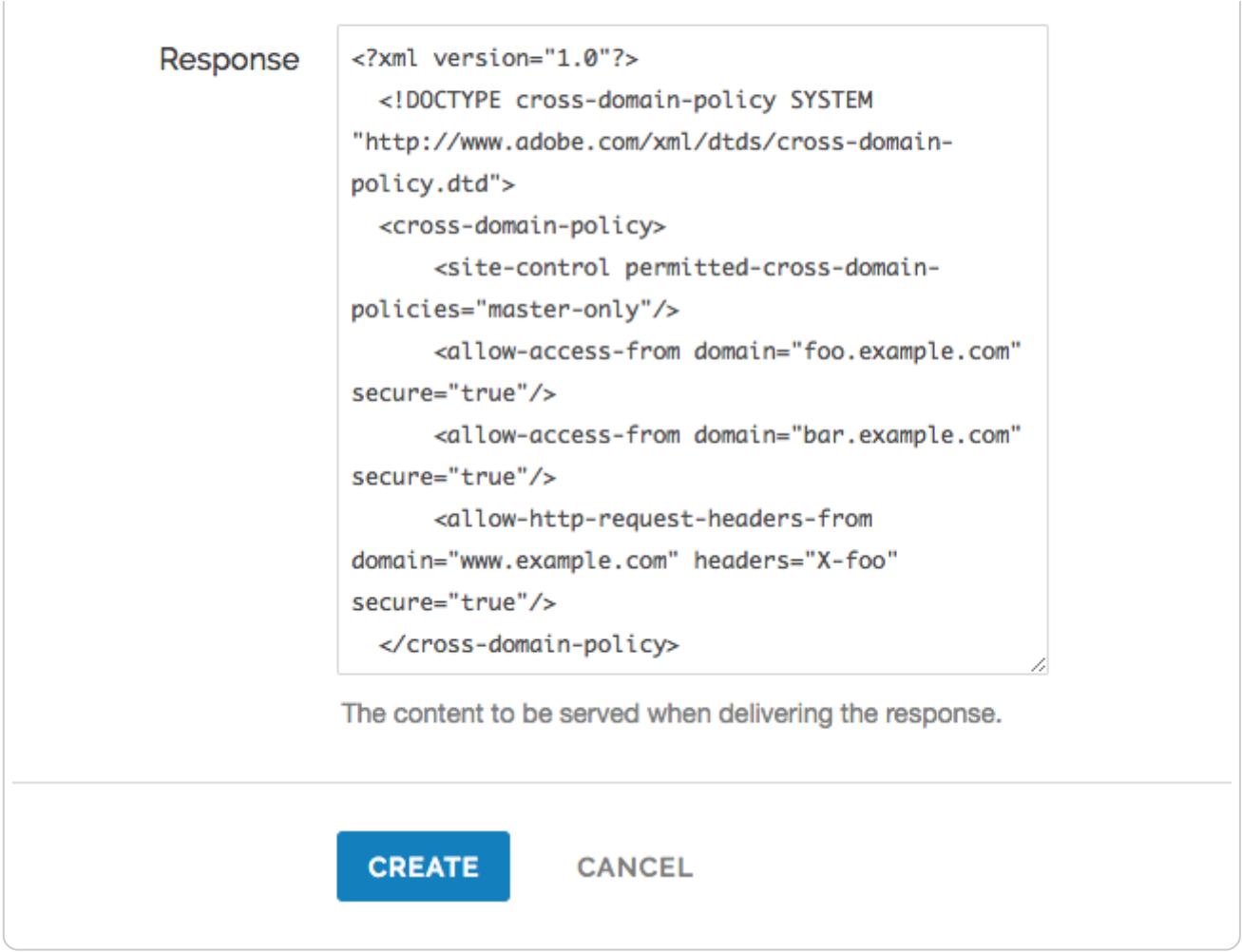
§ Fixing cross-domain errors (/guides/performance-tuning/fixing-cross-domain-errors)

Browser plugins, like Adobe Flash, often require permissions to play content hosted on domains other than from which they are hosted. The crossdomain policy file grants this permission and needs to be present in many cases to allow the content to be played. This guide shows you how to create a synthetic crossdomain.xml response to resolve cross-domain errors.

★ **TIP:** `Error #2048` is a common indicator of a crossdomain.xml issue.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create response** button. The Create a synthetic response page appears.





6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type a human-readable name for the response. For example `crossdomain.xml`.
   - From the **Status** menu, select an HTTP code to return to the client. For example, `200 OK`.
   - In the **MIME Type** field, type `text/x-cross-domain-policy` for the MIME type of the response.
   - In the **Response** field, add the correctly-formatted crossdomain.xml content you want the request to respond with. See cross-domain permissiveness and restrictiveness for additional details.
7. Click the **Create** button. Your new response appears in the list of responses.
8. Click the **Attach a condition** link to the right of the name of your new response. The Create a new condition window appears.
9. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select **Request**.
   - In the **Name** field, type a human-readable name for the response condition. For example `crossdomain.xml`.

- In the **Apply if** field, type `req.url == "/crossdomain.xml"`.

10. Click **Save and apply to** to create the new request condition.
11. Click the **Activate** button to deploy your configuration changes.

Cross-domain permissiveness and restrictiveness

A crossdomain.xml policy file grants these browser plugins permissions to allow content to be played from domains other than that which they are hosted. This file usually has the name `crossdomain.xml` and gets placed by default in the root directory of the domain on which it is hosted. You use this file to define how permissive or restrictive access will be when attempting to play the content being requested.

The following example policy allows the `foo.example.com` and `bar.example.com` domains to pull data, and the `www.example.com` domain to push data via the `X-foo` header:

```
<?xml version="1.0"?>
  <!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
  <cross-domain-policy>
      <site-control permitted-cross-domain-policies="master-only"/>
      <allow-access-from domain="foo.example.com" secure="true"/>
      <allow-access-from domain="bar.example.com" secure="true"/>
      <allow-http-request-headers-from domain="www.example.com" headers="X-foo" secure="true"/>
  </cross-domain-policy>
```

NOTE: This example uses `secure="true"` to force access via HTTPS. You can use `secure="false"` to allow access via HTTP.

Various permissive and restrictive examples of `crossdomain.xml` files appear in Adobe's information on Cross-domain XML for streaming (http://www.adobe.com/devnet/adobe-media-server/articles/cross-domain-xml-for-streaming.html#articlecontentAdobe_numberedheader_3).

---

§ Generating HTTP redirects at the edge (/guides/performance-tuning/generating-http-redirects-at-the-edge)

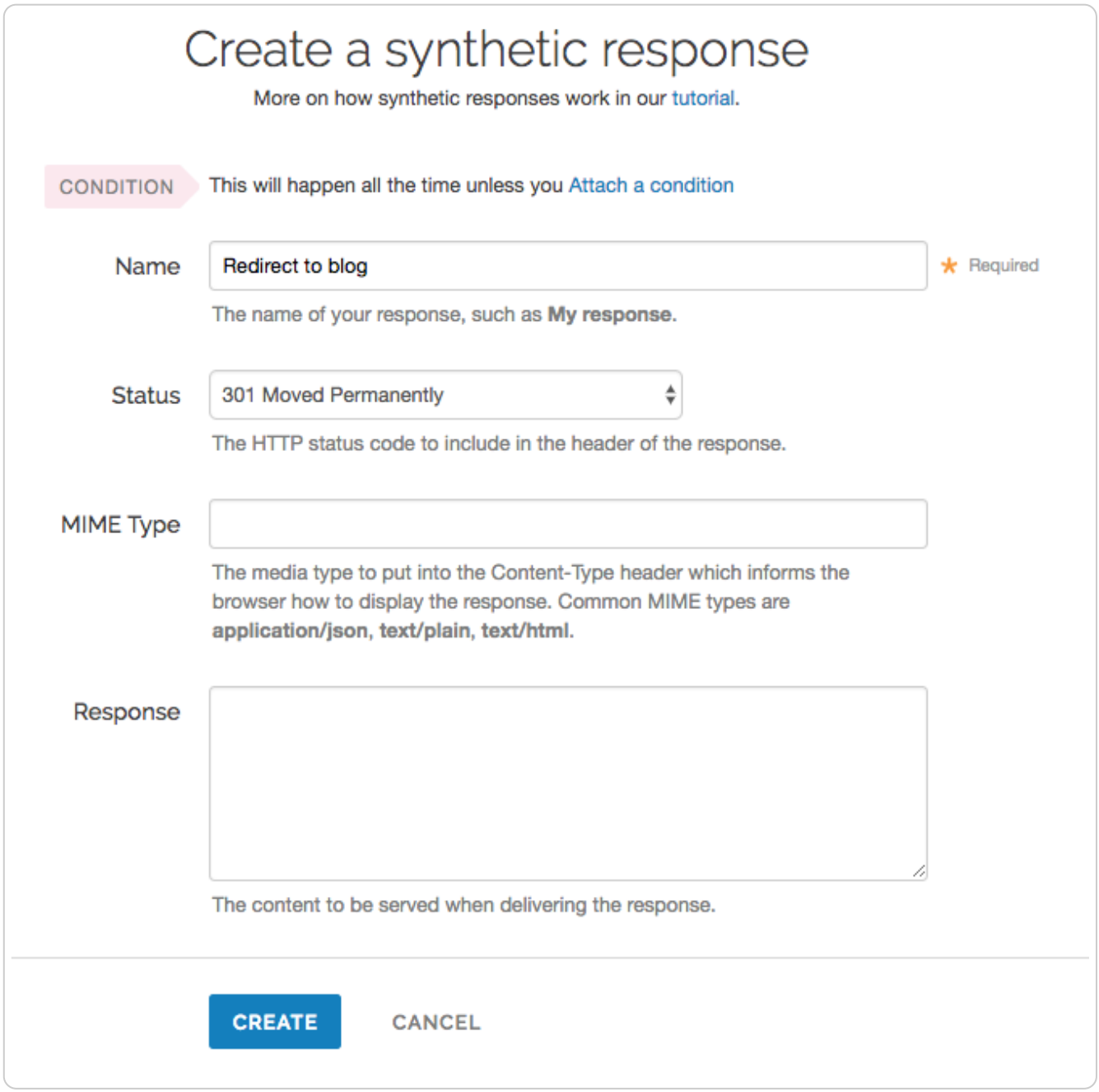
When users request information from your origin servers, you may want to redirect them for various reasons. For example, you may want to redirect them to either a secure or non-secure version of a web page, or to pages that have been moved or updated since the last time they were requested.

You can send these redirects from the edge rather than having to go to origin by creating a synthetic response with the appropriate redirect status code and then creating a content rule with the proper Location header.

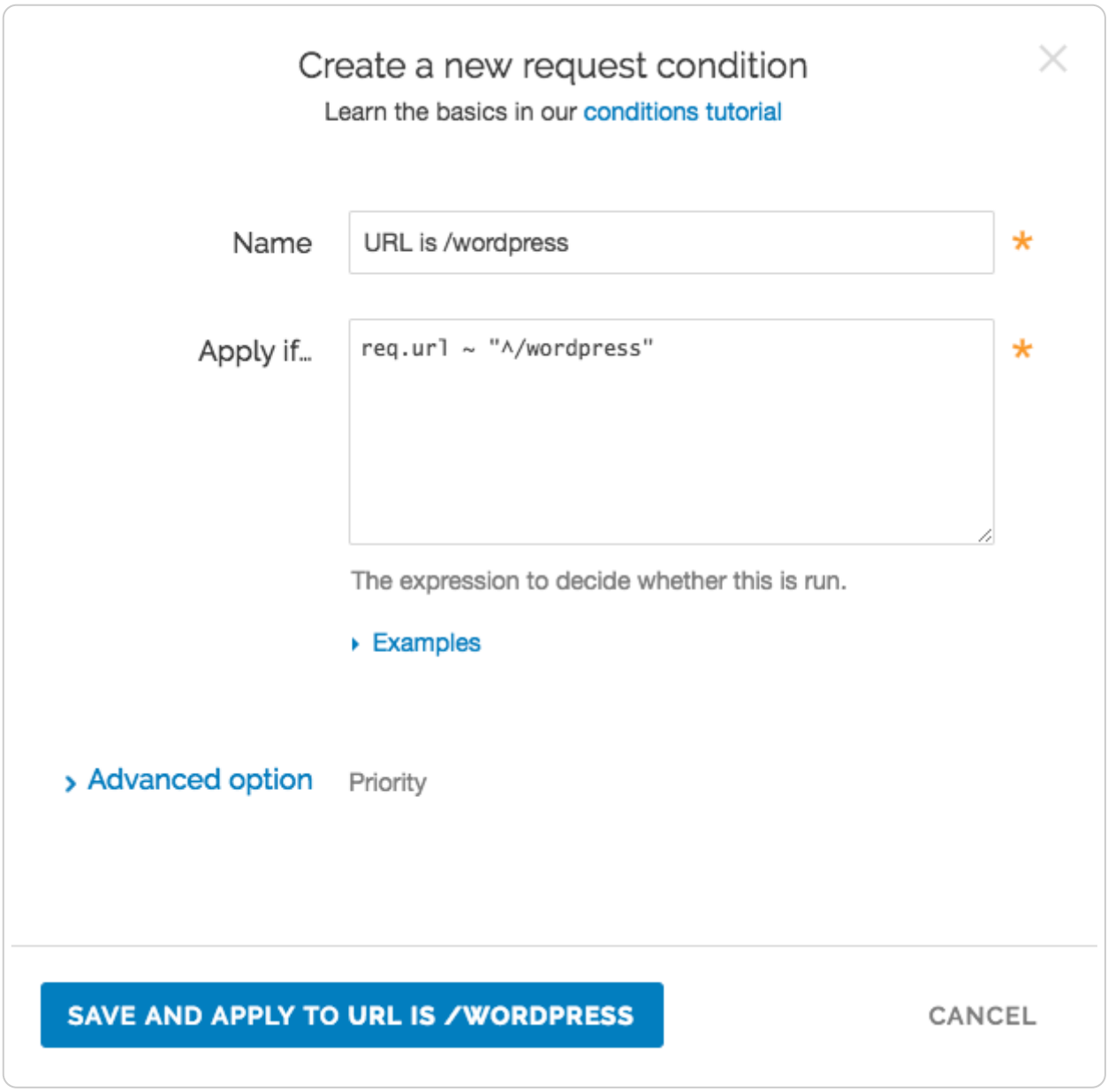
Create a new response and condition

To generate redirects at the edge, start by creating a new response with the appropriate status code and a new condition describing when the response can be applied.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type a meaningful name for your response (e.g., `Redirect to blog`). This name is displayed in the Fastly web interface.
   - From the **Status** menu, select the HTTP status code that should be included in the header of the response (e.g., `301 Moved Permanently` or `302 Moved Temporarily` for redirections).
   - Leave the **MIME Type** field blank.
7. Click the **Create** button to create the new response.
8. On the **Content** page, click the **Attach a condition** link to the right of the new response you just created. The Create a new request condition window appears.

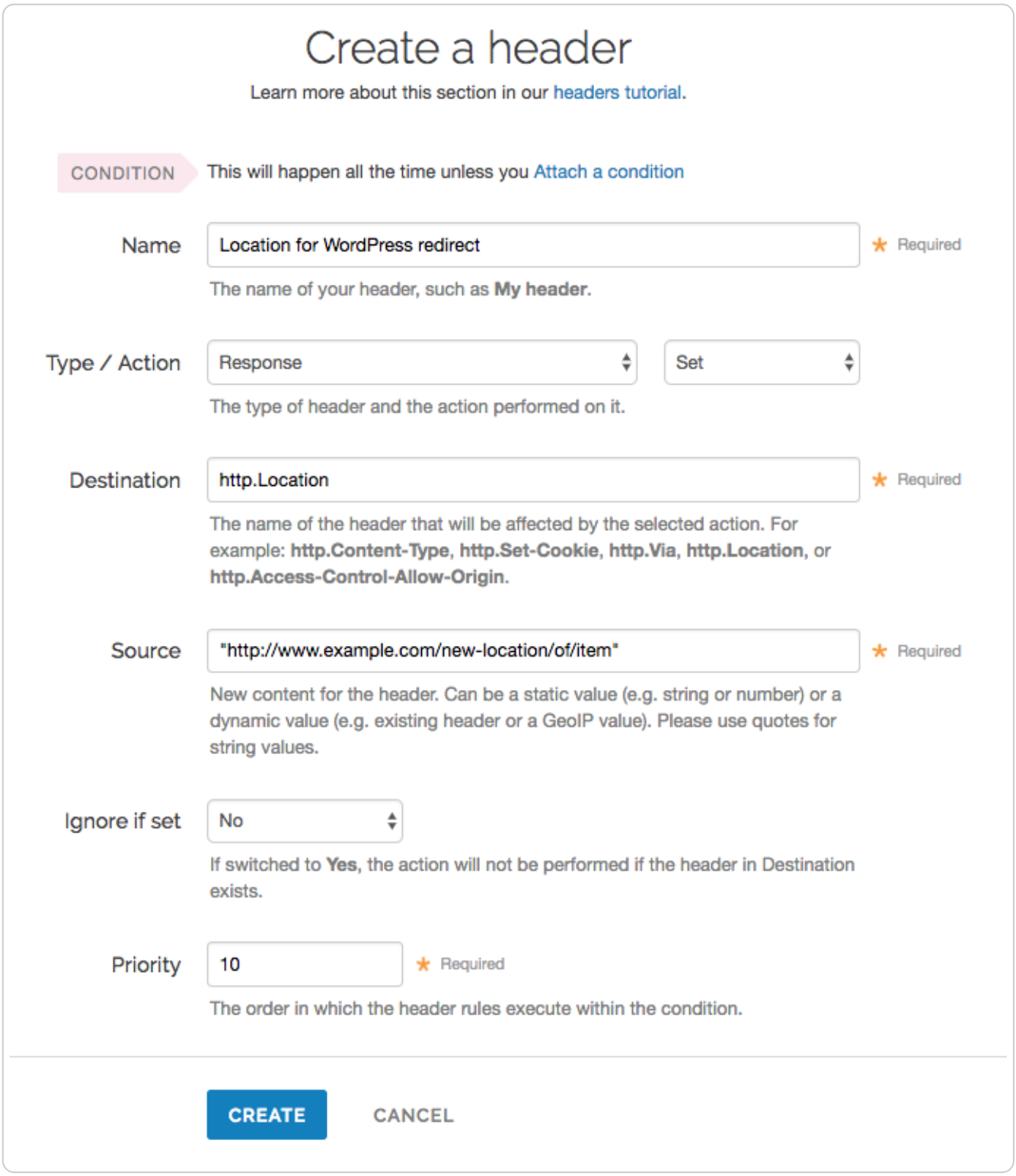


9. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a meaningful name for your condition (e.g., `URL is /wordpress`). This name is displayed in the Fastly web interface.
   - In the **Apply if** field, type the logical expression to execute in VCL to determine if the condition resolves as True or False (e.g., `req.url ~ "^/wordpress"`).
10. Click the **Save and apply to** button to create the new request condition.

Create a new header and condition

Complete the creation of a synthetic redirect by creating a new header and condition that modifies that response by adding the location header based on the status code and the matching URL. This ensures the redirect only applies when both of those are true.

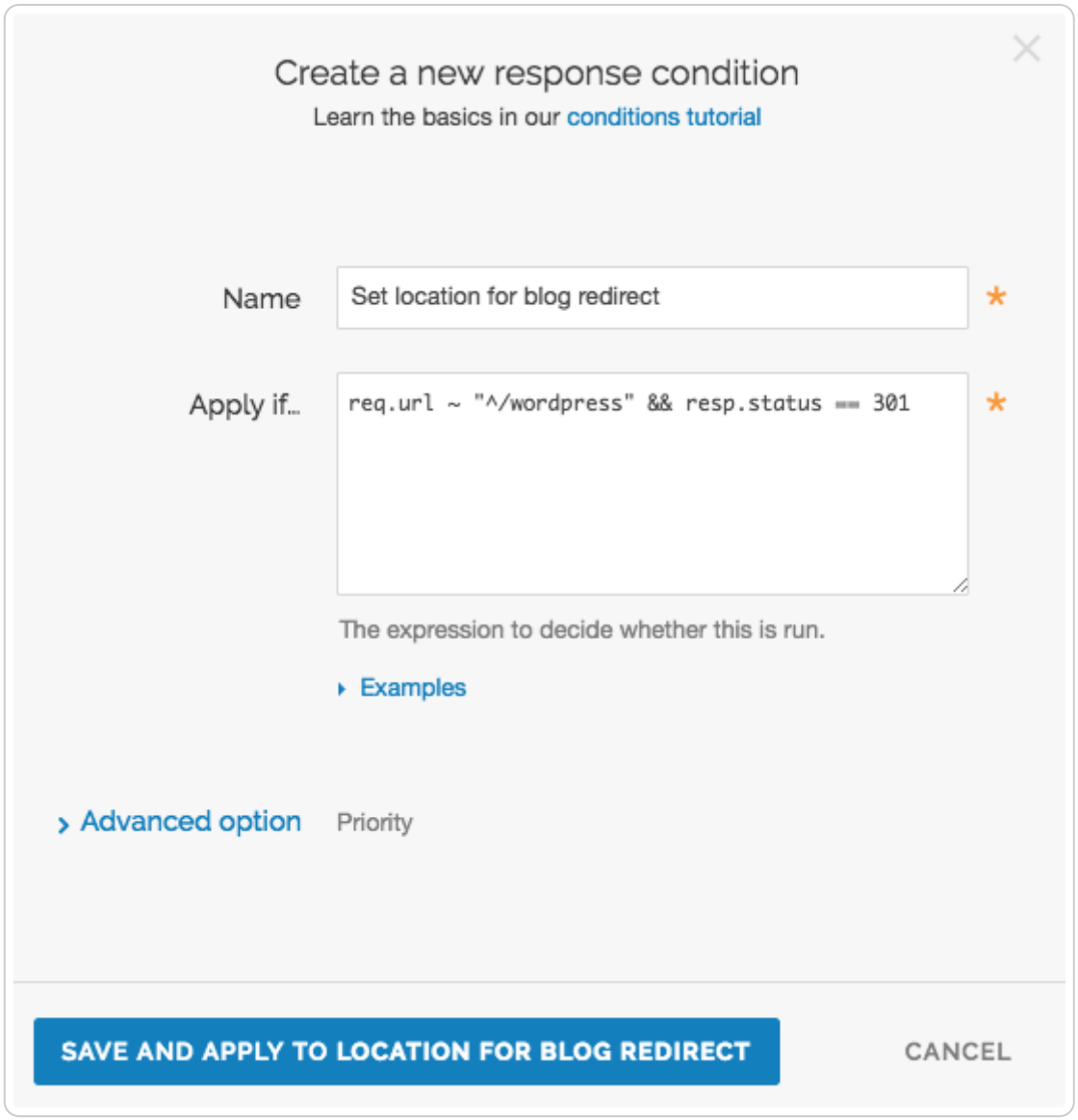
1. On the **Content** page, click the **Create header** button. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a meaningful name for your header (e.g., `Location for WordPress redirect`).
   - From the **Type** menu, select **Response**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Location`.

- In the **Source** field, type the source location of the new content (e.g., `"http://www.example.com/new-location/of/item"`).
- Leave the **Ignore if set** and **Priority** fields at their default settings.

3. Click the **Create** button to create the new header.
4. On the **Content** page, click the **Attach a condition** link to the right of the new header you just created. The Create a new response condition window appears.



5. Fill out the **Create a new response condition** fields as follows:
   - In the **Name** field, type a meaningful name for your condition (e.g., `Set location for blog redirect`).
   - In the **Apply if** field, type the logical expression to execute in VCL to determine if the condition resolves as true or false (e.g., `req.url ~ "^/wordpress" && resp.status == 301`). The `resp.status` needs to match the response code generated in the response above.

6. Click the **Save and apply to** button to create the new condition.
7. Click the **Activate** button to deploy your configuration changes.

NOTE: These responses use a custom status number >500. They will appear as errors on the Dashboard page (/guides/basic-concepts/about-the-web-interface-controls#about-the-dashboard-page) even though they are desired behavior.

§ HTTP/2 server push (/guides/performance-tuning/http2-server-push)

Server push with the `link` response header

Fastly recognizes `link` headers with the preload keyword (https://w3c.github.io/preload/) sent by an origin server and pushes the designated resource to a client. For example, this `link` response header triggers an HTTP/2 push:

```
link: </assets/jquery.js>; rel=preload; as=script
```

We support multiple `link` headers and multiple assets in one `link` header:

```
link: </assets/jquery.js>; rel=preload; as=script, </assets/base.css>; rel=preload; as=style
```

Additional attributes used in the `link` header can further control server push and how the header itself is handled. If no additional attributes are included, the `link` header will trigger server push and be forwarded to the client:

```
link: </assets/jquery.js>; rel=preload; as=script
```

If used with the `nopush` directive, the header will *not* trigger a push and will be passed as is to the client:

```
link: </assets/jquery.js>; rel=preload; as=script; nopush
```

If used with the `x-http2-push-only` directive, the header will trigger a server push but will be subsequently removed and not forwarded to the client:

```
link: </assets/jquery.js>; rel=preload; as=script; x-http2-push-only
```

The attributes can be mixed and matched if needed:

```
link: </assets/jquery.js>; rel=preload; as=script, </assets/base.css>; rel=preload; as=style; n
opush, </assets/main.css>; rel=preload; as=style; x-http2-push-only
```

Link headers and Amazon S3 buckets

If you're using an Amazon Simple Storage Service (S3) (/guides/integrations/amazon-s3) bucket as your origin server, you can still use `link` headers by applying a cache setting condition (/guides/performance-tuning/controlling-caching) like this one:

```
set beresp.http.Link = beresp.http.x-amz-meta-Link
```

Server push with the `h2.push()` function

Server push can also be triggered with the `h2.push()` VCL function. The asset to be pushed is passed to the function as a parameter. For example:

```
sub vcl_recv {
#FASTLY recv

  if (fastly_info.is_h2 && req.url ~ "^/index.html")
  {
    h2.push("/assets/jquery.js");
  }
}
```

The `h2.push()` function triggers server push as soon as it's called, which removes the need for a `link` header to arrive with a server response. This means assets can be pushed to the client before the response for the request that triggered the push is received from the server, accelerating their delivery.

---

§ Improving caching performance with large files (/guides/performance-tuning/improving-caching-performance-with-large-files)

Fastly provides two features to enhance performance specifically for large files up to 5GB: Streaming Miss and Large File Support.

Streaming Miss

When fetching an object from the origin, Streaming Miss ensures the response is streamed back to the client immediately and is written to cache only after the whole object has been fetched. This reduces the first-byte latency, which is the time that the client must wait before it starts receiving the response body. The larger the response body, the more pronounced the benefit of streaming.

> ℹ **NOTE:** </p>

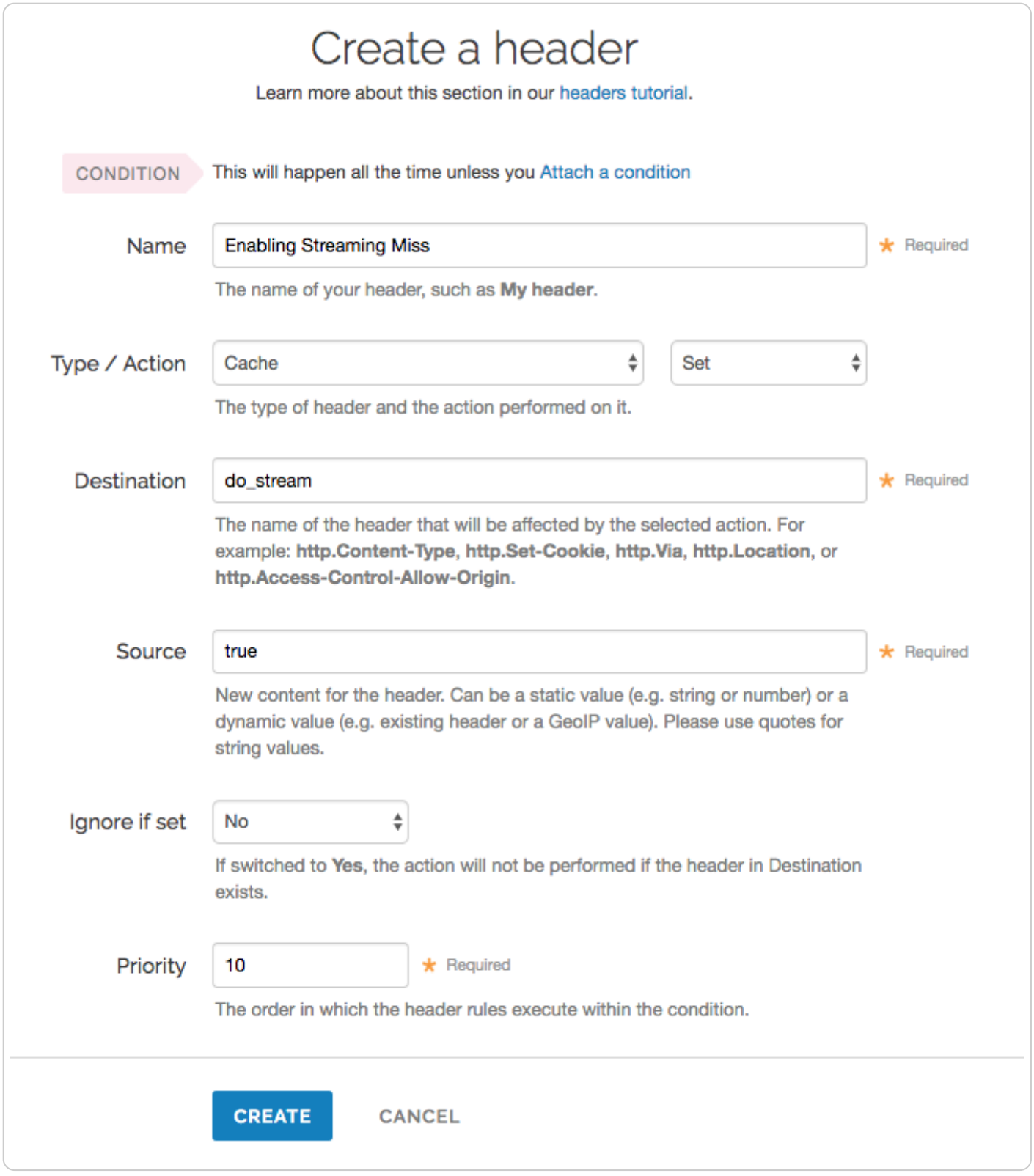
If you enable Streaming Miss, be aware that if an error occurs while transferring the response body, Fastly cannot send an error because the headers are already sent to the client. All we can do is close the connection, truncating the response.

Configuration

Configuration is simple. In VCL, simply set `beresp.do_stream` to true in `vcl_fetch`:

```
sub vcl_fetch {
  ...
  set beresp.do_stream = true;
  ...
  return(deliver);
}
```

The same can be achieved by creating a new header of type `Cache`, action `Set`, destination `do_stream` and source `true` (this can, of course, be controlled with conditions (/guides/conditions/)).



Limitations

There are several limitations to using Streaming Miss.

Origins cannot use TLS

Streaming Miss does not currently support HTTPS (TLS) origin servers. The content can be served to the client over HTTPS, but it cannot be fetched with Streaming Miss over HTTPS. Objects fetched from HTTPS origins are therefore limited to the non-Streaming Miss size of 2GB.

Streaming Miss is not available to HTTP/1.0 clients

If an HTTP/1.0 request triggers a fetch, and the response header from the origin does not contain a Content-Length field, then Streaming Miss will be disabled for the fetch and the fetched object will be subject to the non-streaming-miss object size limit.

If an HTTP/1.0 request is received while a Streaming Miss for an object is in progress, the HTTP/1.0 request will wait for the response body to be downloaded before it will receive the response header and the response body, as if the object was being fetched without Streaming Miss.

Cache hits are not affected. An HTTP/1.0 client can receive a large object served from cache, just like an HTTP/1.1 client.

Streaming Miss is not compatible with on-the-fly gzip compressing of the fetched object

Streaming Miss can handle large files whether or not they are compressed. However, on-the-fly compression of objects that are not already compressed is not compatible with Streaming Miss. If the VCL sets `beresp.gzip` to true, Streaming Miss will be disabled.

Streaming Miss is not compatible with ESI (Edge-Side Includes)

Responses that are processed through ESI cannot be streamed. Responses that are included from an ESI template cannot be streamed. When ESI is enabled for the response or when the response is fetched using `<esi:include>`, then Streaming Miss will be disabled and the fetched object will be subject to the non-streaming-miss object size limit of 2GB.

Large File Support

Large File Support is automatically enabled for all clients — there's no need to manually configure anything. You should, however, be aware that there are maximum file sizes and several failure modes.

Maximum file size

If Streaming Miss is enabled, then the maximum size is slightly below 5GB (specifically 5,368,578,048 bytes). With Streaming Miss disabled, the maximum size is still higher than the previous maximum, but is limited to a little under 2GB (specifically 2,147,352,576 bytes).

Failure modes

There are several failure modes you may encounter while using Large File Support.

What happens when the maximum object size limit is exceeded?

If the response from the origin has a Content-Length header field which exceeds the maximum object size, Fastly will immediately generate a 503 response to the client unless specific VCL is put in place to act on the error.

If no Content-Length header field is returned, Fastly will start to fetch the response body. If while fetching the response body we determine that the object exceeds maximum object size, we will generate a status 503 response to the client (again, unless specific VCL is in place to act on the error).

If no Content-Length header field is present and Streaming Miss is in effect, Fastly will stream the content back to the client. However, if while streaming the response body Fastly determines that the object exceeds the maximum object size, it will terminate the client connection abruptly. The client will detect a protocol violation, because it will see its connection close without a properly terminating 0-length chunk.

What happens when an origin read fails?

A failure to read the response header from the origin, regardless of Streaming Miss, causes a 503 response (which can be acted on in VCL).

If reading the response body from the origin fails or times out, the problem will be reported differently depending on whether Streaming Miss is in effect for the fetch. Without Streaming Miss, a 503 response will be generated. With Streaming Miss, however, it is already too late to send an error response since the header will already have been sent. In this case, Fastly will again abruptly terminate the client connection and the client will detect a protocol violation. If the response was chunked, it will see its connection close without a properly terminating 0-length chunk. If Content-Length was known, it will see the connection close before the number of bytes given.

Incidentally, this is the reason why HTTP/1.0 clients cannot be supported by Streaming Miss in the cases when the Content-Length is not yet known or available. Without the client receiving a Content-Length and without support for chunking, the client cannot distinguish the proper end of the download from an abrupt connection breakage anywhere upstream from it.

---

§ Load-balancing configuration (/guides/performance-tuning/load-balancing-configuration)

This guide describes how to automatically load balance between two or more origin servers. Load balancing distributes requests across multiple servers to optimize resource use and avoid overloading any single resource.

Before you begin

Before you configure load balancing, keep in mind the following:

- To prevent errors when shielding is enabled (/guides/performance-tuning/shielding#enabling-shielding), all backends in the automatic load balancing group must use the same shielding location.
- Conditions on your origin server can directly change how automatic load balancing behaves. Be sure to review conditions behavior to ensure automatic load balancing works properly.
- Many customers configure failover at the same time they configure load balancing functionality. Our guide on configuring failover (/guides/performance-tuning/failover-configuration) can show you how.

Enabling load balancing

To enable round-robin load balancing across two or more origin servers, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the name of the host you want to edit. The Edit this host page appears.
6. From the **Auto load balance** menu, select **Yes**.
7. In the **Weight** field, type the percentage of the total traffic to send to the origin server.

   ★ **TIP:** When you specify a whole number in the **Weight** field, you specify the percentage of the total traffic to send to a specific origin server. Each origin server receives the percentage (<weight>/<total>) of the total traffic equal to the number you specify. For example, if you have two origin servers, A and B, setting the weight to 50 on both splits the traffic between them equally. Each origin server receives 50 percent of your total traffic. If you increase the weight on origin server A to 55 and decrease the weight on origin server B to 45, the percentage of traffic changes to 55 percent and 45 percent respectively.

8. Click the **Update** button.
9. Repeat steps 5, 6, 7, and 8 for each origin server you want to include in the automatic load balancing group.

> ℹ **NOTE:** Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com (mailto:sales@fastly.com) to enable more than five origin servers per service in your account.

10. Click the **Activate** button to deploy your configuration changes.

Using conditions with load balancing

You can set conditions (/guides/conditions/) on origin servers or headers to change how load balancing works.

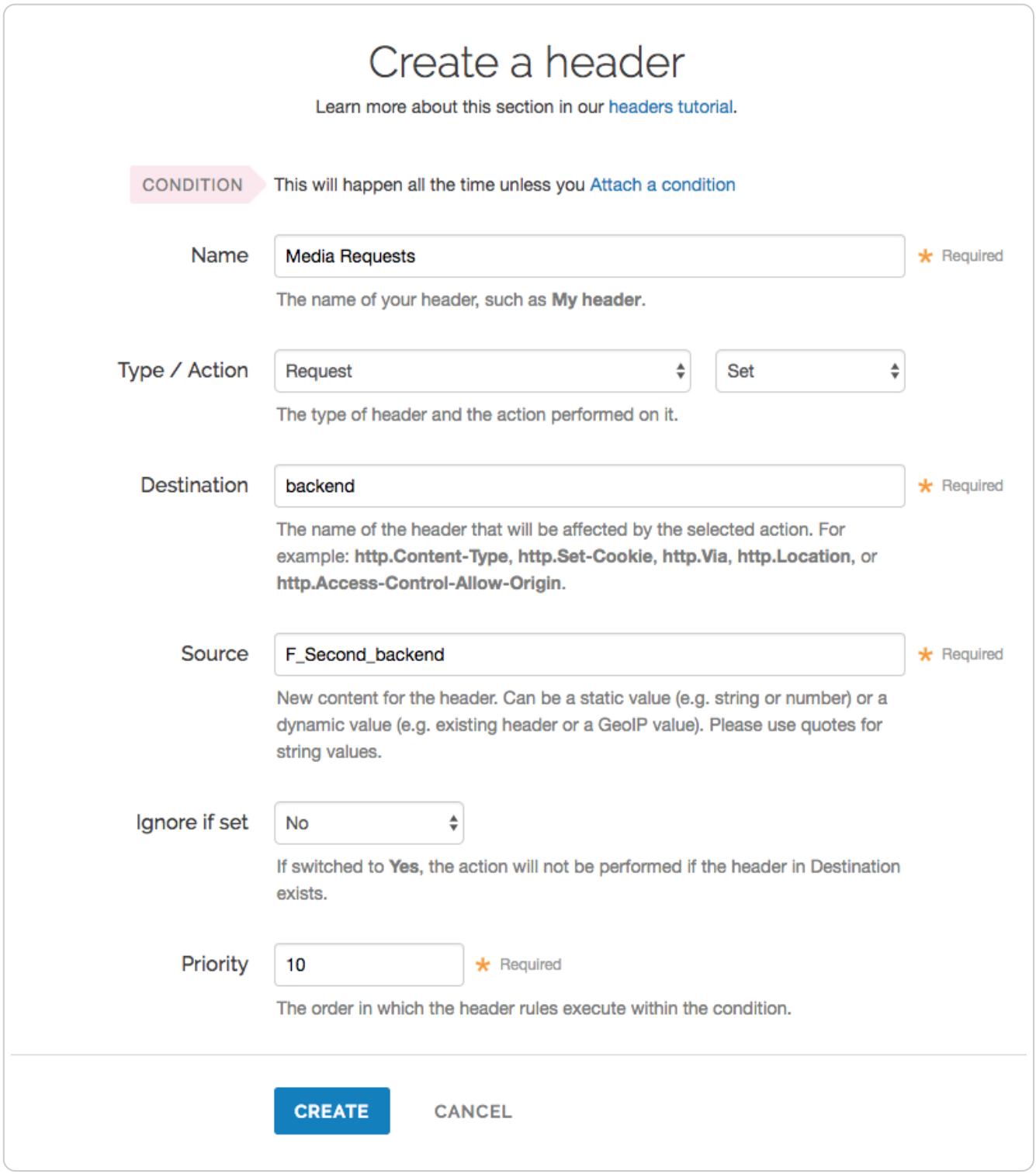
Setting conditions on origin servers

When you set conditions on origin servers, you can potentially change how automatic load balancing works. The load balancing autodirector groups servers together based on like conditions, giving you the flexibility to effectively create subsets of the autodirector by assigning a condition to one group of origins and another condition to another set of origins. If each group of origin servers has a different condition that affects load balancing, the auto load function will not randomly load balance between the different servers.

Setting conditions on headers

Conditions can also be assigned to a server through a header. For example, you have three servers called F_Fastly, F_Second_backend, and F_Third_backend and want all URLs with a certain prefix to default to the second server. First, you'd create a header.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. In the **Headers** area, click the **Create header** button to create a new header. The Create a header page appears.



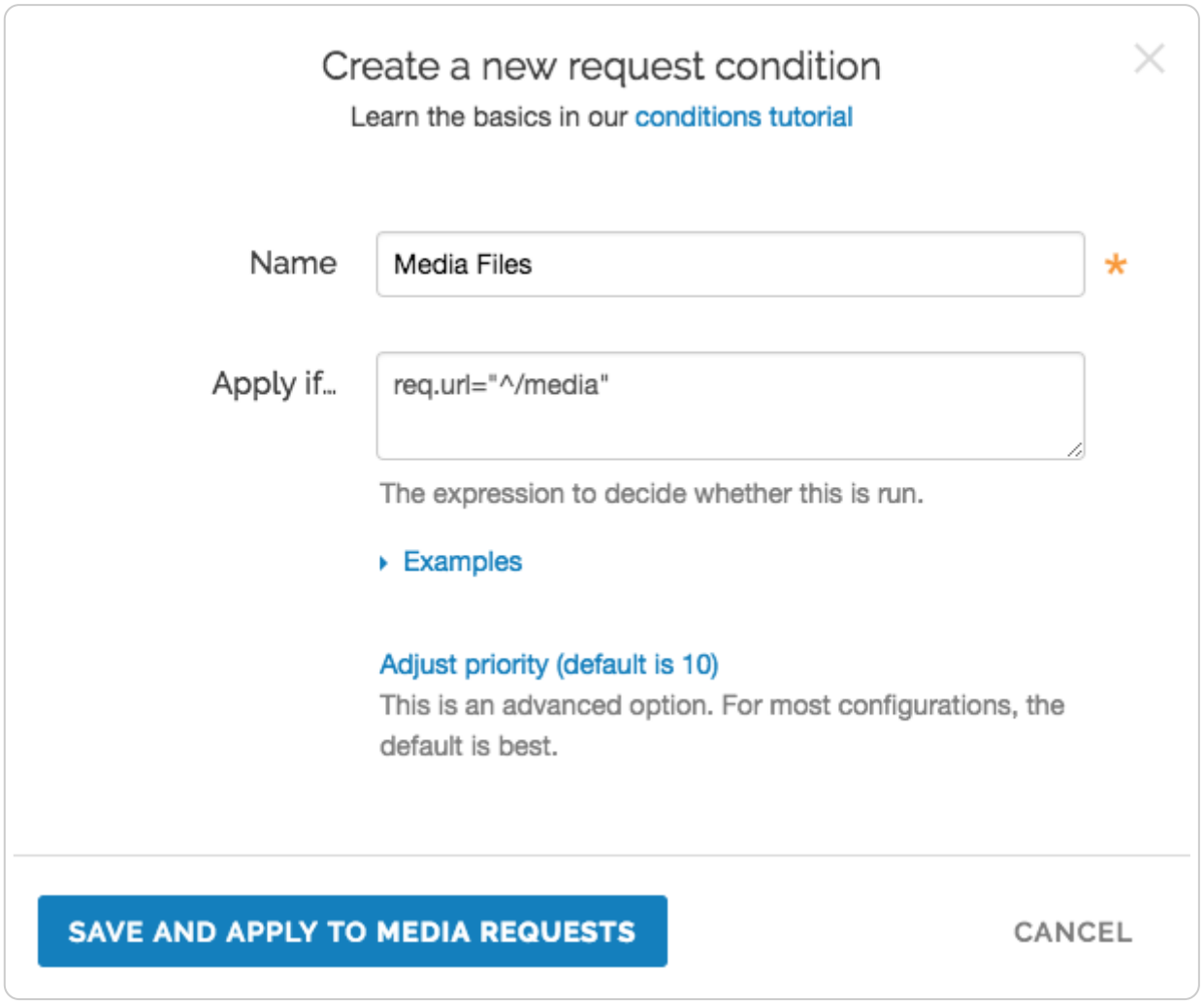
6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the new header (for example, `Media Requests`).
   - From the **Type** menu, select **Request** and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header that will be affected by the action (for example, `backend`).

- In the **Source** field, type the name of the origin server the content for this header comes from (for example, `F_Second_backend`).
- Leave the **Ignore if set** and **Priority** fields set to their default settings.

7. Click **Create**.

After the header is created, you'd create a new condition to apply if the URL matches the desired prefix.

1. In the **Headers** area, click the **Attach a condition** link next to the name of the new header you just created. The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Media Files`).
   - In the **Apply if** field, type the request condition that will be applied (for example, `req.url="^/media"`).
   - Leave the priority set to its default value.
3. Click the **Save and apply to** button to create the new condition for the header.

4. Click the **Activate** button to deploy your configuration changes.

The generated VCL below illustrates the autodirector set for all three servers. Within the section **sub vcl_recv**, the default origin server is set to the autodirector and, if the media condition is met, requests are forwarded to the second server. If the condition is not met, requests are forwarded to one of the three servers at random.

```
director autodirector_ random {
  {
    .backend = F_Second_backend;
    .weight  = 100;
  }{
    .backend = F_Third_backend;
    .weight  = 100;
  }{
    .backend = F_Fastly;
    .weight  = 100;
  }
}

sub vcl_recv {
#--FASTLY RECV CODE START
  if (req.restarts == 0) {
    if (!req.http.X-Timer) {
      set req.http.X-Timer = "S" time.start.sec "." time.start.usec_frac;
    }
    set req.http.X-Timer = req.http.X-Timer ",VS0";
  }

  # default conditions
  set req.backend = autodirector_;

  # end default conditions

  # Request Condition: Media files Prio: 10
  if (req.url ~ "^/media") {

    # Header rewrite Media Requests : 10
    set req.backend = F_Second_backend;
  }
  #end condition

#--FASTLY RECV CODE END
}
```

§ Maintaining separate HTTP and HTTPS requests to origin servers

(/guides/performance-tuning/maintaining-separate-http-and-https-requests-to-backend-servers)

It is common to use the same origin web application to serve both HTTP and HTTPS requests and let the application determine which actions to take to secure communications (/guides/securing-communications/) depending on the incoming protocol. Fastly allows users to set this up to preserve this functionality within their servers. To set Fastly up to send HTTP requests to the non-secure service and HTTPS requests to the secure service, configure two origins, one each for the secure and non-secure ports, then set up the conditions under which requests will be sent there.

Create multiple origins

Begin by configuring the same origin address with a different port as a separate origin server. Follow the instructions for connecting to origins (/guides/basic-configuration/connecting-to-origins). You'll add specific details about the non-secure server (port `80`) when you fill out the **Create a host** fields:

- In the **Name** field, type a name for the non-secure server (for example, `Server Name (plain)`).
- In the **Address** field, type the address of the non-secure server (for example, `server.example.com`).
- In the **Transport Layer Security (TLS)** section, set **Enable TLS?** to **No**.

Follow the instructions for connecting to origins (/guides/basic-configuration/connecting-to-origins) to create another origin server, this time for your secure server. You'll add specific details about the secure server (port `443`) when you fill out the **Create a host** fields:

- In the **Name** field, type a name for the non-secure server (for example, `Server Name (secure)`).
- In the **Address** field, type the address of the non-secure server (for example, `server.example.com`).
- In the **Transport Layer Security (TLS)** section, leave the **Enable TLS?** default set to **Yes**.

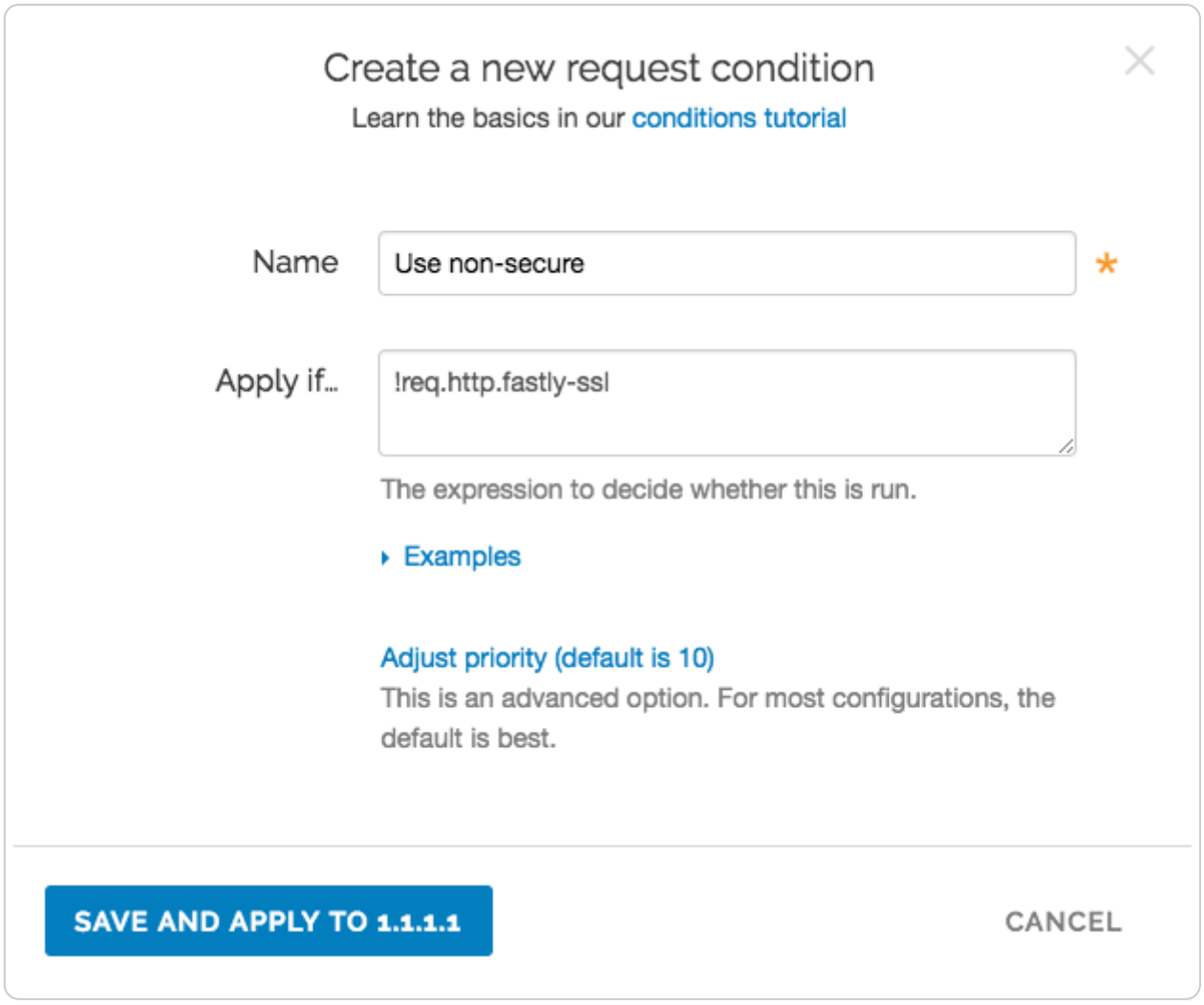
Conditionally send traffic to origins

To conditionally determine which server receives secure and non-secure requests, Fastly relies on the presence or absence of a specific header when the backend is selected. When an incoming connection is received over TLS, Fastly sets the `req.http.fastly-ssl` header to determine which

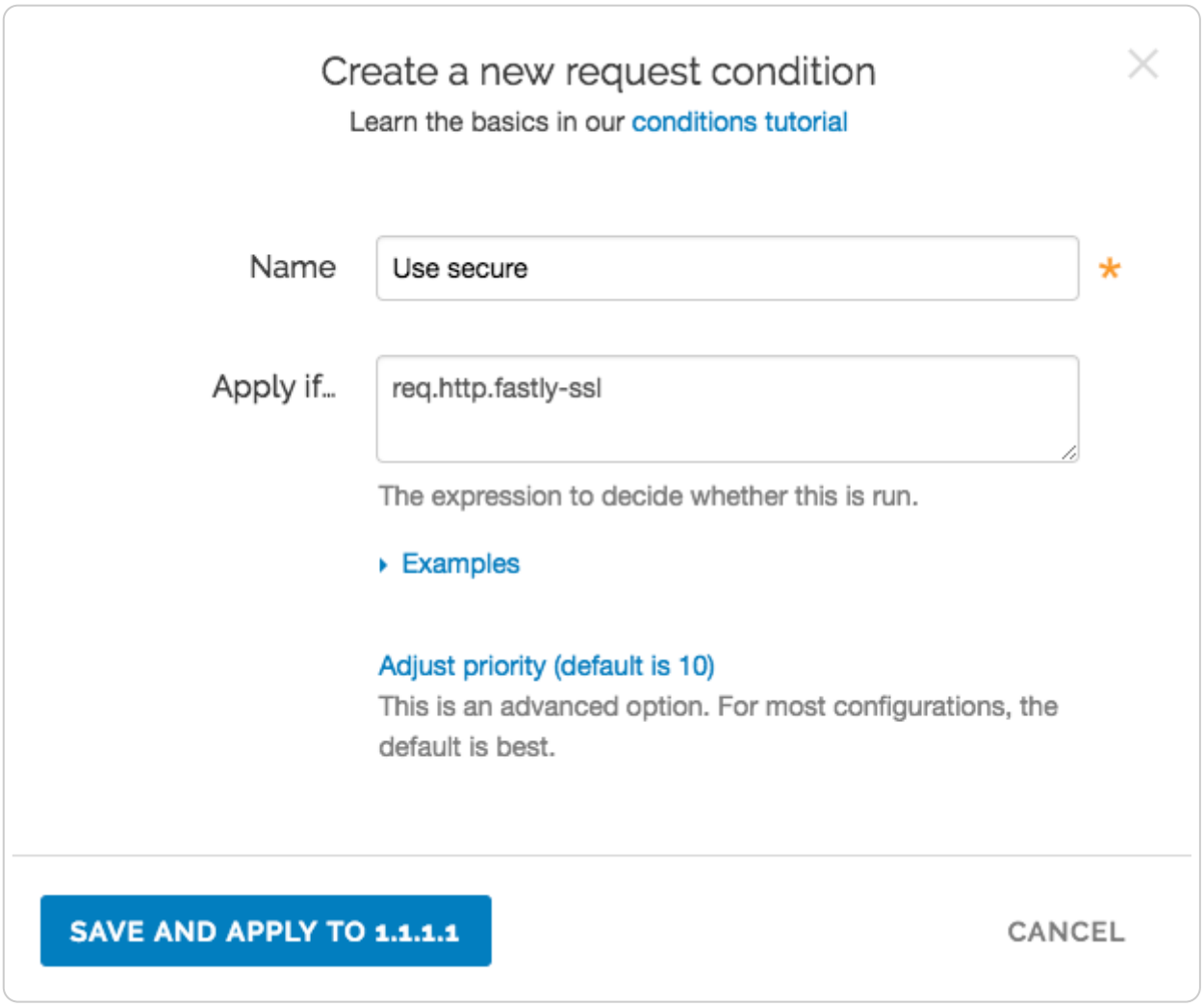
server to use.

Set a condition for this header on each origin by following the steps below.

1. On the **Origins** page, click the **Attach a condition** link next to the name of the non-secure server. The Create a new request condition window appears.



2. Fill out the **Create a new request** fields as follows:
   - In the **Name** field, type the name of the condition specifying use of the non-secure server (for example, `Use non-secure`).
   - In the **Apply if** field, type `!req.http.fastly-ssl`.
   - Leave the priority set to its default value.
3. Click the **Save and apply to** button to create the new condition.
4. On the **Origins** page, click the **Attach a condition** link next to the name of the secure server. The Create a new request condition window appears.



5. Fill out the **Create a new request condition** window as follows:
   - In the **Name** field, type the name of the condition specifying use of the secure server (for example, `Use secure`).
   - In the **Apply if** field, type `req.http.fastly-ssl`.
   - Leave the priority set to its default value.
6. Click the **Save and apply to** button to create the new condition.
7. Click the **Activate** button to deploy your configuration changes.

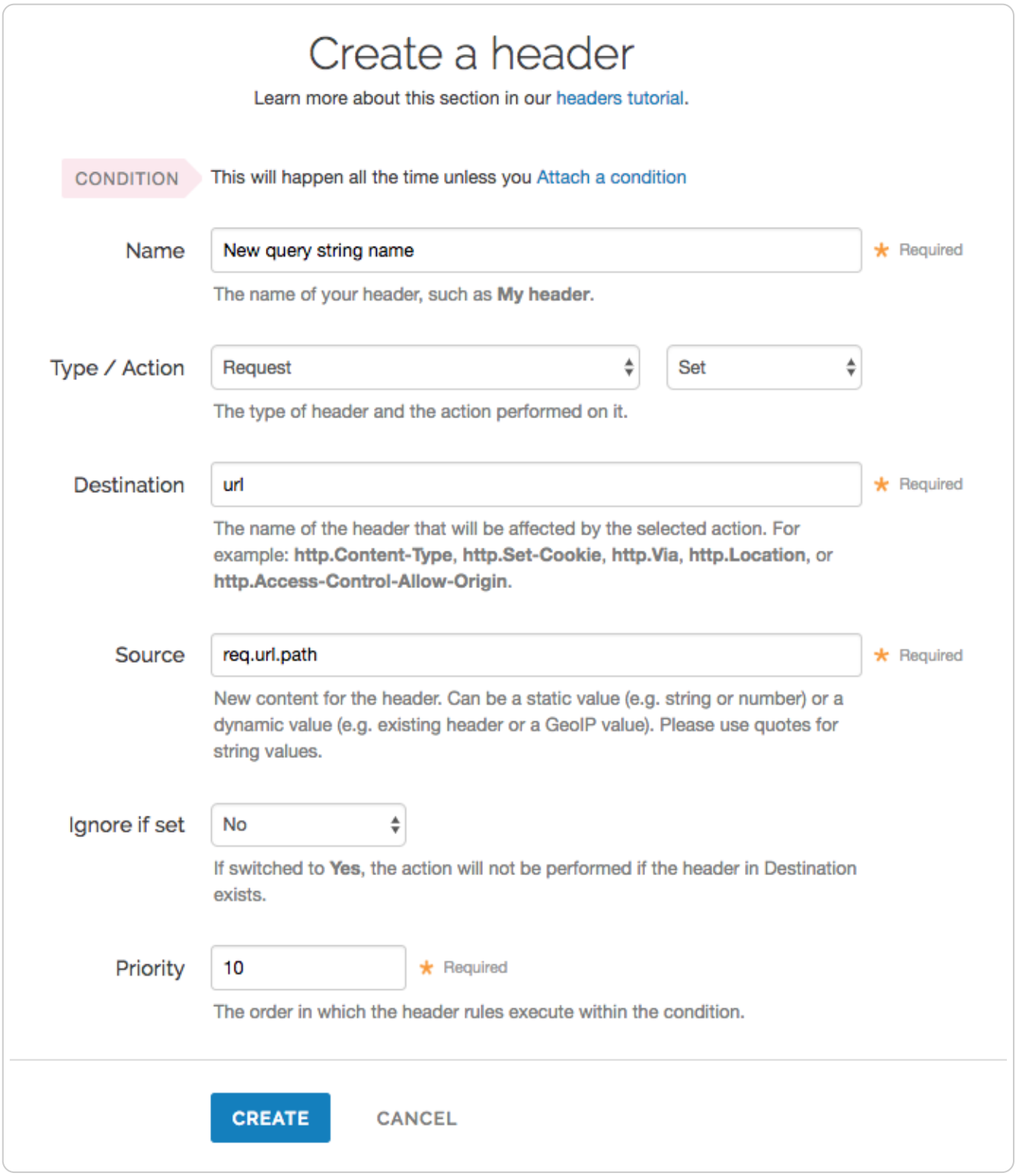
§ Making query strings agnostic (/guides/performance-tuning/making-query-strings-agnostic)

Under normal circumstances, Fastly would consider these URLs different objects that are cached separately:

- http://example.com
- http://example.com?asdf=asdf
- http://example.com?asdf=zxcv

It is possible, however, to have them all ignore the query string and return the same cached file.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a description for the header (e.g., `New query string name`).
   - From the **Type** menu, select **Request**, and from **Action** menu, select **Set**.
   - In the **Destination** field, type `url`.
   - In the **Source** field, type `req.url.path`.
   - From the **Ignore if set** menu, select **No**.

- Set the **Priority** field to whatever priority you want.

7. Click the **Create** button to create the new header. The new header you created appears on the Content page.
8. Click the **Activate** button to deploy your configuration changes.

The request will be sent to the origin as a URL without the query string.

For more information about controlling caching, see our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

---

§ Request collapsing (/guides/performance-tuning/request-collapsing)

This guide describes Fastly's Request Collapsing feature, frequently used when creating advanced service configurations.

> ℹ **NOTE:** This guide requires advanced knowledge of Varnish and the VCL language (/guides/vcl/guide-to-vcl).

The basics

Request Collapsing causes simultaneous cache misses within a single Fastly datacenter to be "collapsed" into a single request to an origin server. While the single request is being processed by the origin, the other requests wait in a queue for it to complete. Two types of Request Collapsing exist:

1. Collapsing on a single cache server
2. Collapsing within the datacenter between cache servers

Each cache server will automatically queue duplicate requests for the same hash and only allow one request to origin. You can disable this behavior by setting `req.hash_ignore_busy` to `true` in `vcl_recv`.

Within a datacenter, not every cache stores every object. Only two servers in each datacenter will store an object: one as a primary and one as a backup. Only those two servers will fetch the object from origin.

How it works

In Fastly's version of Varnish, VCL subroutines often run on different caches during a request. For a particular request, both an edge cache and a shield cache will exist (though a single cache can, in some cases, fulfill both of these roles). The edge cache receives the HTTP request from the client and determines via a hash which server in the datacenter is the shield cache. If this cache determines it is the shield cache and has the object in cache, it fulfills both the edge cache and the shield cache roles.

Certain VCL subroutines run on the edge cache and some on the shield cache:

- Edge Cache: `vcl_recv`, `vcl_deliver`, `vcl_log`, `vcl_error`
- Shield Cache: `vcl_miss`, `vcl_hit`, `vcl_pass`, `vcl_fetch`, `vcl_error`

Determining if a cache is an edge or a shield

The `fastly_info.is_cluster_edge` VCL variable will be true if the cache currently running the VCL is the edge cache and false if it is the shield cache.

Caveats

Keep in mind the following limitations when using the Request Collapsing feature:

1. Any `req.http.*` headers are not transferred from the shield cache back to the edge cache. Remember this when writing advanced configurations that use headers to keep track of state. If you set a `req.http.*` header in any of the subroutines that run on the shield cache, expect that the change will not persist on the edge cache.
2. A single, slow request to origin can sometimes cause a great many other requests for the same object to hang and fail. Because many requests for a single object are being collapsed down to one, they all succeed or fail based on the request that reaches the origin.

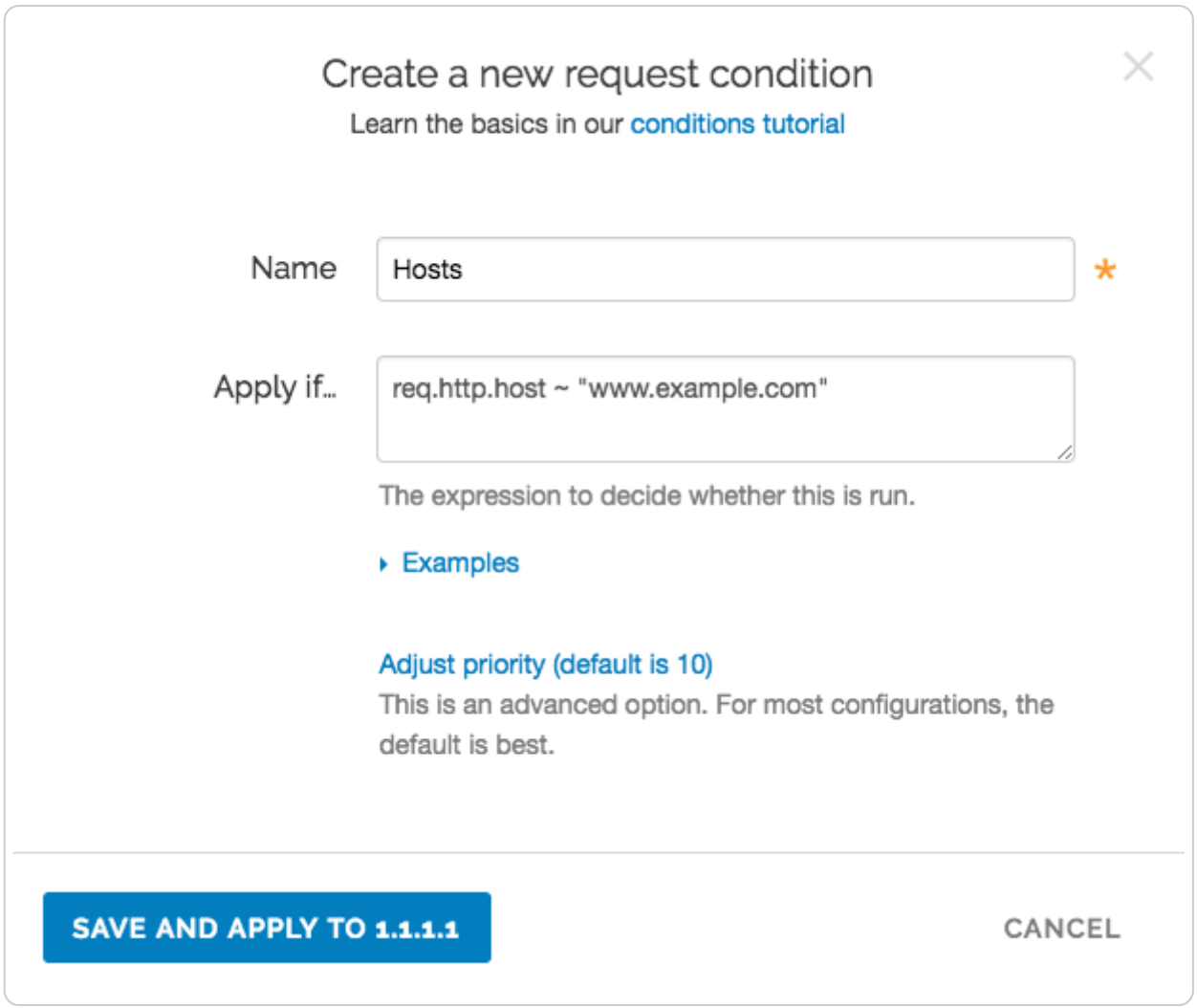
---

§ Routing assets to different origins (/guides/performance-tuning/routing-assets-to-different-origins)

Some customers have assets stored on multiple origin servers and want to route various requests to specific, different servers based on criteria they supply (e.g., asset type, file directory, host header). Fastly offers customers the ability to set conditions on their origins, which simply adds an if statement block to your VCL.

Basic setup: Create conditions for each origin

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **Attach a condition** link to the right of the name of an origin server. The Create a new request condition window appears.

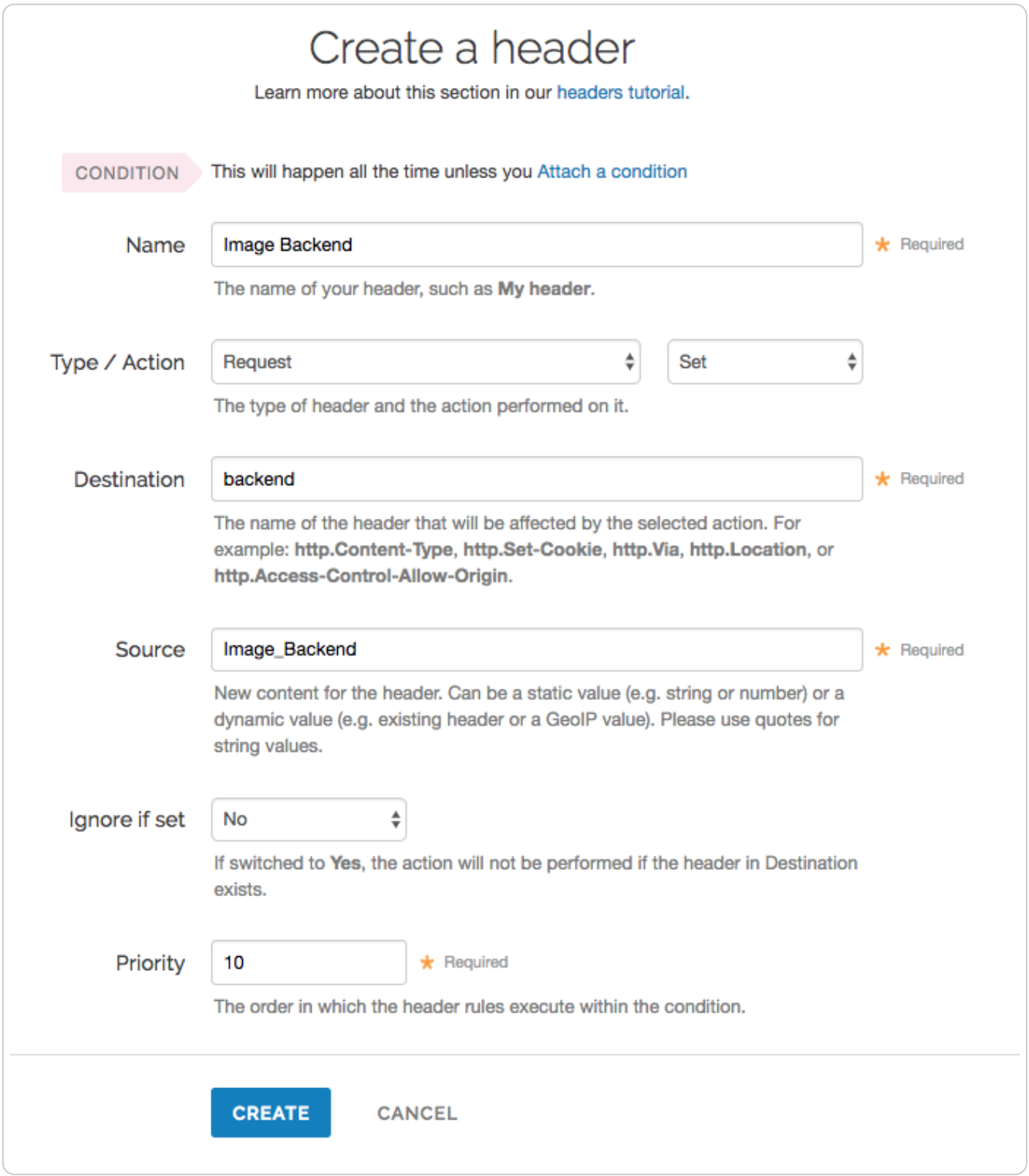


6. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a human-readable name for the condition.
   - In the **Apply if** field, type the conditions that you want to apply to your origin server. For example, for hosts, you could type `req.http.host ~ "www.example.com"`. Or, for content-type / URL, you could type `req.url ~ ".(jpg|png|gif)($|\?)"`.
7. Click the **Save and apply to** button. The new condition appears on the Origins page.
8. Click the **Activate** button to deploy your configuration changes.

Backup setup: Create a header

What if you have a condition already assigned to your origin? Although you can group request conditions on the origin with an 'and' or 'or' clause, there can only ever be one condition rule attached to that origin. If you want to separate your request conditions instead of grouping them, you can use header rules to route assets to different origins instead.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.

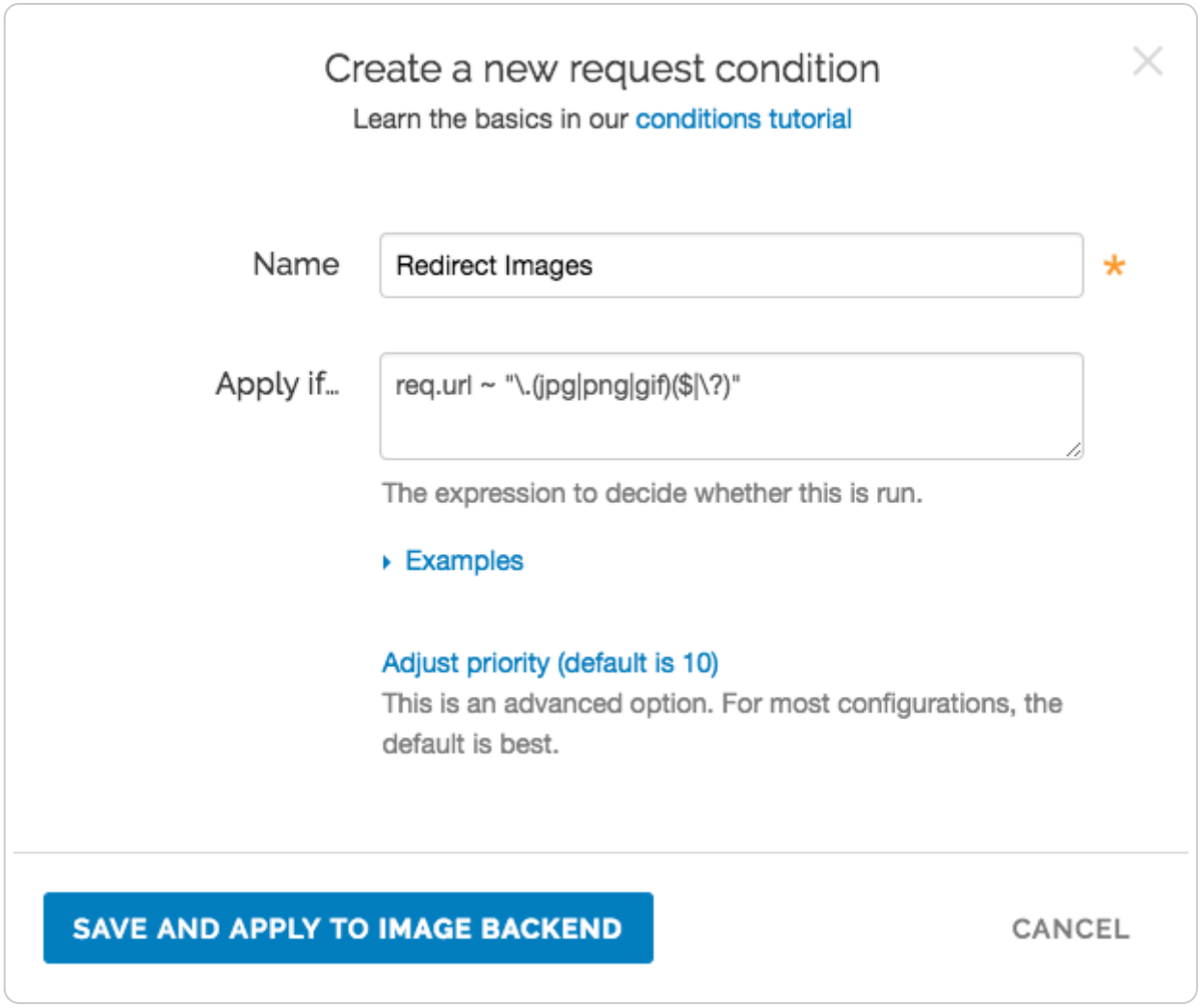


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Image Backend` (or any meaningful, preferred name).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `backend`.
   - In the **Source** field, type `Image_Backend`. (This should match the name of your global origin server. You can see the exact name if you look at your VCL. Click on the **VCL**

button at the top of the page.)

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button. The new header appears on the Content page.
8. On the **Content** page, click the **Attach a condition** link next to the header you just created. The Create a new request condition window appears.



9. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type `Redirect Images` (or any meaningful, preferred name).
   - In the **Apply if** field, type `req.url ~ "\.(jpg|png|gif)($|\?)"`.
10. Click the **Save and apply to** button. The condition appears on the Content page.
11. Click the **Activate** button to deploy your configuration changes.

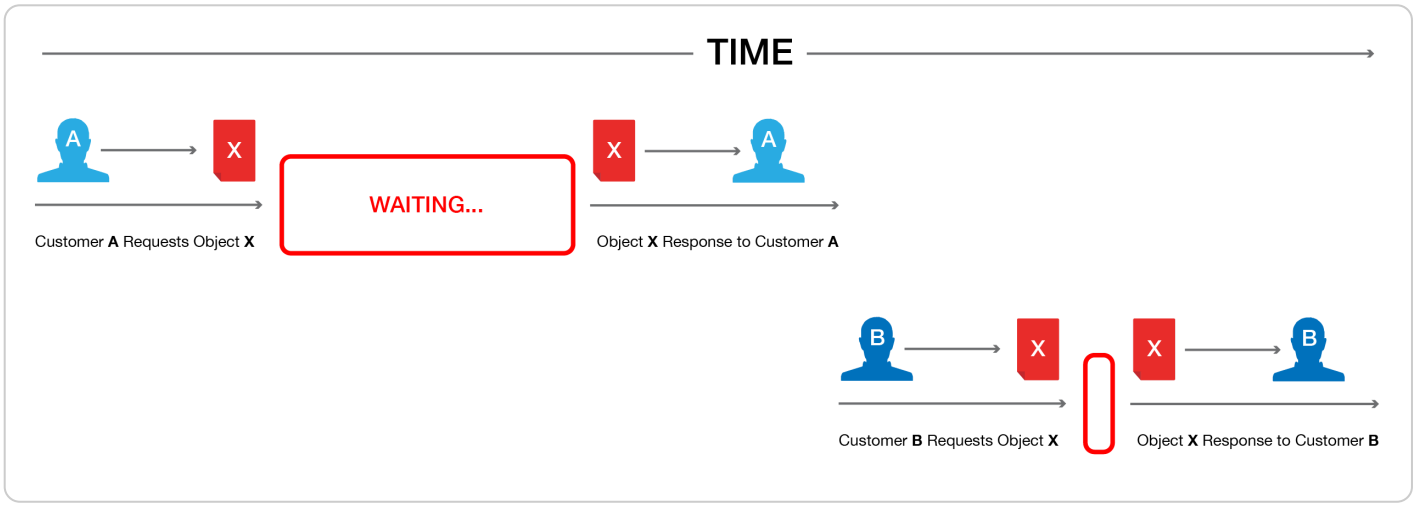
★ **TIP:** Learn more about conditions in our Conditions subcategory (/guides/conditions/).

§ Serving stale content (/guides/performance-tuning/serving-stale-content)

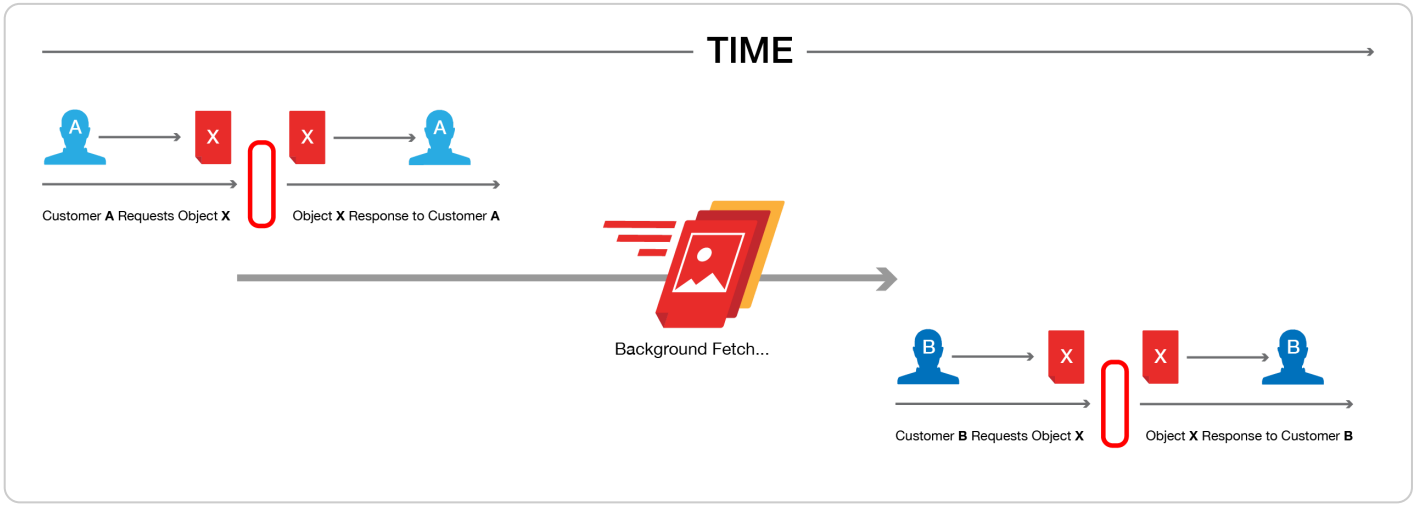
Fastly can optionally serve stale content when there is a problem with your origin server, or if new content is taking a long time to fetch from your origin server. For example, if Fastly can't contact your origin server, our POPs (/guides/basic-concepts/fastly-pop-locations) will continue to serve cached content when users request it. These features are not enabled by default.

Serving old content while fetching new content

Certain pieces of content can take a long time to generate. Once the content is cached it will be served quickly, but the first user to try and access it will pay a penalty.



This is unavoidable if the cache is completely cold, but if this is happening when the object is in cache and its TTL is expired, then Fastly can be configured to show the stale content while the new content is fetched in the background.



Fastly builds on the behavior proposed in RFC 5861 (http://tools.ietf.org/html/rfc5861) "HTTP Cache-Control Extensions for Stale Content" by Mark Nottingham, which is under consideration for inclusion in Google's Chrome browser (https://www.mnot.net/blog/2014/06/01/chrome_and_stale-while-revalidate).

Usage

To activate this behavior simply add a `stale-while-revalidate` or `stale-if-error` statement to either the Cache-Control or Surrogate-Control headers in the response from your origin server. For example:

```
Cache-Control: max-age=600, stale-while-revalidate=30
```

will cache some content for 10 minutes and, at the end of that 10 minutes, will serve stale content for up to 30 seconds while new content is being fetched.

Similarly, this statement:

```
Surrogate-Control: max-age=3600, stale-if-error=86400
```

instructs the cache to update the content every hour (3600 seconds) but if the origin is down then show stale content for a day (86400 seconds).

Alternatively, these behaviors can be controlled from within VCL by setting the following variables in vcl_fetch:

```
set beresp.stale_while_revalidate = 30s;
```

```
set beresp.stale_if_error = 86400s;
```

Interaction with grace

Stale-if-error works exactly the same as Varnish's grace variable such that these two statements are equivalent:

```
set beresp.grace = 86400s;
```

```
set beresp.stale_if_error = 86400s;
```

However, if a grace statement is present in VCL it will override any `stale-while-revalidate` or `stale-if-error` statements in any Cache-Control or Surrogate-Control response headers.

Setting `beresp.stale_if_error` either via header or via VCL does nothing on its own. In order to serve stale, follow the instructions below.

Serving stale content on errors

In certain situations where your origin server becomes unavailable, you may want to serve stale content. These instructions provide an advanced configuration that allows all three possible origin failure cases to be handled using VCL. These instructions require the ability to upload custom VCL.

> ❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

In the context of Varnish, there are three ways an origin can fail:

- The origin can be marked as unhealthy by failing health checks.
- If Varnish cannot contact the origin for any reason, a 503 error will be generated.
- The origin returns a valid HTTP response, but that response is not one we wish to serve to users (for instance, a 503).

The custom VCL shown below handles all three cases. If the origin is unhealthy, the default serve stale behavior is triggered by `stale-if-error`. In between the origin failing and being marked unhealthy, Varnish would normally return 503s. The custom VCL allows us to instead either serve stale if we have a stale copy, or to return a synthetic error page. The error page can be customized. The third case is handled by intercepting all 5XX errors in `vcl_fetch` and either serving stale or serving the synthetic error page.

> ⚠ **WARNING:** Do not purge all (/guides/purging/single-purges#purging-all-content) cached content if you are seeing 503 errors (/guides/debugging/common-503-errors). Purge all overrides stale-if-error and increases the requests to your origin server, which could result in additional 503 errors.

Although not strictly necessary, health checks (/guides/basic-configuration/health-checks-tutorial) should be enabled in conjunction with this VCL. Without health checks enabled, all of the functionality will still work, but serving stale or synthetic responses will take much longer while waiting for an origin to timeout. With health checks enabled, this problem is averted by the origin being marked as unhealthy.

The custom VCL shown below includes the Fastly standard boilerplate. Before uploading this to your service, be sure to customize or remove the following values to suit your specific needs:

- `if (beresp.status >= 500 && beresp.status < 600)` should be changed to include any HTTP response codes you wish to serve stale/synthetic for.
- `set beresp.stale_if_error = 86400s;` controls how long content will be eligible to be served stale and should be set to a meaningful amount for your configuration. If you're

sending `stale_if_error` in Surrogate-Control or Cache-Control from origin, remove this entire line.

- `set beresp.stale_while_revalidate = 60s;` controls how long the `stale_while_revalidate` feature will be enabled for an object and should be set to a meaningful amount for your configuration. This feature causes Varnish to serve stale on a cache miss and fetch the newest version of the object from origin in the background. This can result in large performance gains on objects with short TTLs, and in general on any cache miss. Note that `stale_while_revalidate` overrides `stale_if_error`. That is, as long as the object is eligible to be served stale while revalidating, `stale_if_error` will have no effect. If you're sending `stale_while_revalidate` in Surrogate-Control or Cache-Control from origin, remove this entire line.
- `synthetic {"<!DOCTYPE html>Your HTML!</html>"};` is the synthetic response Varnish will return if no stale version of an object is available and should be set appropriately for your configuration. You can embed your HTML, CSS, or JS here. Use caution when referencing external CSS and JS documents. If your origin is offline they may be unavailable as well.

```
sub vcl_recv {
  if (req.http.Fastly-FF) {
    set req.max_stale_while_revalidate = 0s;
  }

#FASTLY recv

  if (req.request != "HEAD" && req.request != "GET" && req.request != "FASTLYPURGE") {
    return(pass);
  }

  return(lookup);
}

sub vcl_fetch {
  /* handle 5XX (or any other unwanted status code) */
  if (beresp.status >= 500 && beresp.status < 600) {

    /* deliver stale if the object is available */
    if (stale.exists) {
      return(deliver_stale);
    }

    if (req.restarts < 1 && (req.request == "GET" || req.request == "HEAD")) {
      restart;
    }

    /* else go to vcl_error to deliver a synthetic */
    error 503;
  }

  /* set stale_if_error and stale_while_revalidate (customize these values) */
  set beresp.stale_if_error = 86400s;
  set beresp.stale_while_revalidate = 60s;

#FASTLY fetch

  if ((beresp.status == 500 || beresp.status == 503) && req.restarts < 1 && (req.request == "GE
T" || req.request == "HEAD")) {
    restart;
  }

  if (req.restarts > 0) {
    set beresp.http.Fastly-Restarts = req.restarts;
  }

  if (beresp.http.Set-Cookie) {
    set req.http.Fastly-Cachetype = "SETCOOKIE";
    return(pass);
  }
```

```
  if (beresp.http.Cache-Control ~ "private") {
    set req.http.Fastly-Cachetype = "PRIVATE";
    return(pass);
  }

  /* this code will never be run, commented out for clarity */
  /* if (beresp.status == 500 || beresp.status == 503) {
     set req.http.Fastly-Cachetype = "ERROR";
     set beresp.ttl = 1s;
     set beresp.grace = 5s;
     return(deliver);
  } */

  if (beresp.http.Expires || beresp.http.Surrogate-Control ~ "max-age" || beresp.http.Cache-Con
trol ~ "(s-maxage|max-age)") {
    # keep the ttl here
  } else {
    # apply the default ttl
    set beresp.ttl = 3600s;
  }

  return(deliver);
}

sub vcl_hit {
#FASTLY hit

  if (!obj.cacheable) {
    return(pass);
  }
  return(deliver);
}

sub vcl_miss {
#FASTLY miss
  return(fetch);
}

sub vcl_deliver {
  if (resp.status >= 500 && resp.status < 600) {
    /* restart if the stale object is available */
    if (stale.exists) {
      restart;
    }
  }

#FASTLY deliver
  return(deliver);
}

sub vcl_error {
```

```
#FASTLY error

  /* handle 503s */
  if (obj.status >= 500 && obj.status < 600) {

    /* deliver stale object if it is available */
    if (stale.exists) {
      return(deliver_stale);
    }

    /* otherwise, return a synthetic */

    /* include your HTML response here */
    synthetic {"<!DOCTYPE html><html>Replace this text with the error page you would like to se
rve to clients if your origin is offline.</html>"};
    return(deliver);
  }

}

sub vcl_pass {
#FASTLY pass
}

sub vcl_log {
#FASTLY log
}
```

Why serving stale content may not work as expected

Here are some things to consider if Fastly isn't serving stale content:

- **Cache:** Stale objects are only available for cacheable content.
- **Shielding:** If you don't have shielding (/guides/performance-tuning/shielding) enabled, a POP (/guides/basic-concepts/fastly-pop-locations) can only serve stale on errors if a request for that cacheable object was made through that POP before. We recommend enabling shielding to increase the probability that stale content on error exists. Shielding is also a good way to quickly refill the cache after a performing a purge all (/guides/purging/single-purges#purging-all-content).
- **Requests:** As traffic to your site increases, you're more likely to see stale objects available (even if shielding is disabled). It's reasonable to assume that popular assets will be cached at multiple POPs.
- **Least Recently Used (LRU):** Fastly has an LRU list, so objects are not necessarily guaranteed to stay in cache for the entirety of their TTL (time to live). But eviction is

dependent on many factors, including the object's request frequency, its TTL, the POP from which it's being served. For instance, objects with a TTL of longer than 3700s get written to disk, whereas objects with shorter TTLs end up in transient, in-memory-only storage. We recommend setting your TTL to more than 3700s when possible.

- **Purges:** Whenever possible, you should purge content using our soft purge feature (/guides/purging/soft-purges). Soft purge allows you to easily mark content as outdated (stale) instead of permanently deleting it from Fastly's caches. If you can't use soft purge, we recommend purging by URL (/guides/purging/single-purges#purging-a-url) or using surrogate keys (/guides/purging/getting-started-with-surrogate-keys) instead of performing a purge all (/guides/purging/single-purges#purging-all-content).

Additional reading

- RFC 5861 (http://tools.ietf.org/html/rfc5861)
- Mark Nottingham's blog post (https://www.mnot.net/blog/2014/06/01/chrome_and_stale-while-revalidate)
- Chrome discussion (https://groups.google.com/a/chromium.org/forum/#!msg/chromium-dev/zchogDvIYrY/ZqWSdt3LJdMJ)

---

§ Shielding (/guides/performance-tuning/shielding)

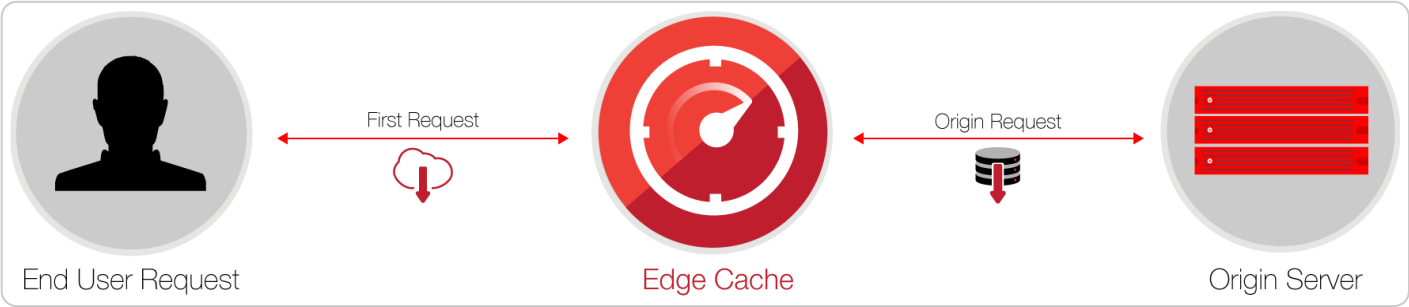
Fastly's shielding service feature (/guides/detailed-product-descriptions/about-fastlys-full-site-delivery-features#origin-shielding) allows you to designate a specific Point of Presence (POP) as a shield node to your origins. Once designated, all requests to your origin will go through that datacenter, increasing the chances of getting a HIT (/guides/performance-tuning/understanding-cache-hit-and-miss-headers-with-shielded-services) for a given resource. If a different POP doesn't have a specific object, it will query the shield (provided it's not down for maintenance) instead of your origins.

How shielding works

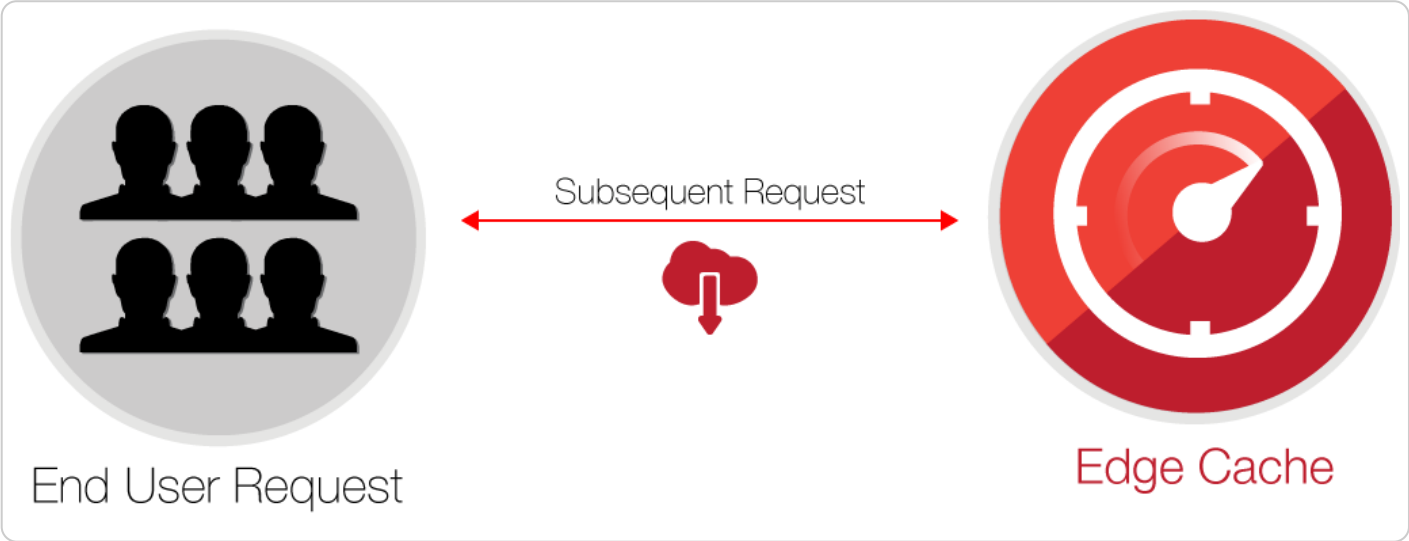
When a user requests brand new content from a customer's server and that content has never been cached by any Fastly POP (/guides/basic-concepts/fastly-pop-locations), this is what happens to their request when shielding is and is not enabled.

Without shielding enabled

Without shielding enabled, when the first request for content arrives at POP A, the POP does not have the content cached. It passes the request along to a customer's origin server to get the content. Once the content is retrieved, POP A caches it and sends it on to the user.



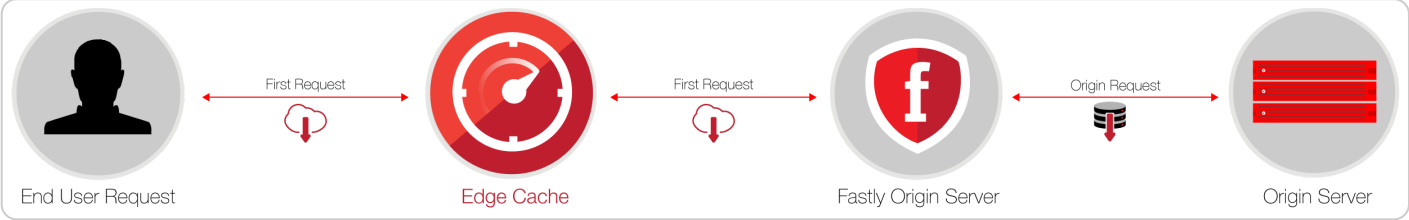
When a second request for that same content arrives at POP A, the content is already cached. No request goes to the customer's origin server. It's merely sent from the cached copy.



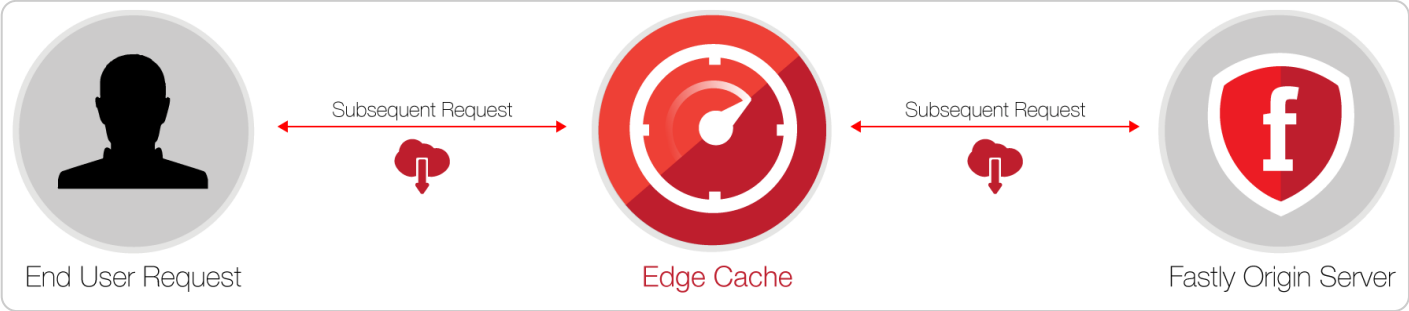
If the second request were to arrive at POP B instead of POP A, however, the request would once again be passed along to the customer's origin server. It would then be cached and passed back to the end user, just like POP A did when that info was first requested.

With shielding enabled

With shielding enabled (/guides/performance-tuning/shielding), when the first request for content arrives at the POP A, that POP does not have the content cached. It passes the request along to the shield POP, which also doesn't have the content cached. The shield POP passes the request along to the customer's origin server. It then caches the content that's retrieved and passes it along to POP A. POP A then passes the content along to the user.



When a second request for that same content arrives at POP A, the content is already cached, so no request goes to the shield POP or the customer's origin server.



If the second request were to arrive at POP B instead of POP A, however, the request would be passed along to the shield POP. That shield POP already has a cached copy from the first request to POP A. No future requests for the content would be passed along to the customer's origin server until the shield POP's cached copy of it expires.

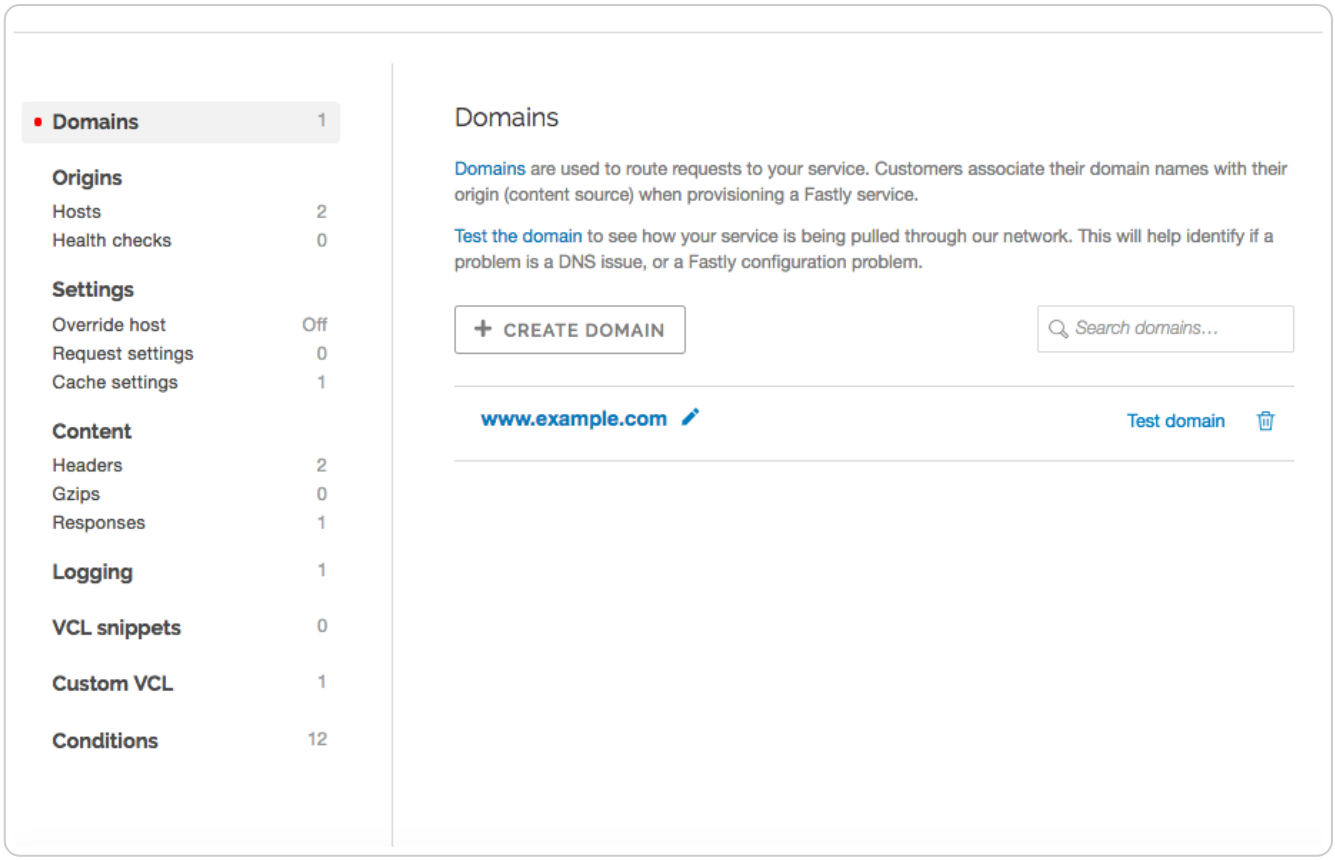
Enabling shielding

> ❗ **IMPORTANT:** If you are using Google Cloud Storage as your origin, you need to follow the steps in our GCS setup guide (/guides/integrations/google-cloud-storage) instead of the steps below.

Enable shielding with these steps:

1. Read the caveats of shielding information below for details about the implications of and potential pitfalls involved with enabling shielding for your organization.
2. Log in to the Fastly web interface and click the **Configure** link.
3. From the service menu, select the appropriate service.
4. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
5. Click the **Origins** link. The Origins page appears.
6. Click the name of the host you want to edit. The Edit this host page appears.
7. From the **Shielding** menu, select the datacenter to use as your shield keeping the following in mind:
    - Generally, we recommend selecting a datacenter close to your backend. Doing this allows faster content delivery because we optimize requests between the shield POP you're selecting (the one close to your server) and the edge POP (the one close to the user making the request).
    - With multiple backends, each backend will have its own shield defined. This allows flexibility if your company has backends selected geographically and different shield POPs are desired.

8. Click **Update** to save your changes.
9. If you have changed the default host or have added a header to change the host, add the modified hostname to your list of domains. Do this by clicking the **Domains** link and checking to make sure the host in question appears on the page. If it isn't included, add it by clicking the **Create domain** button.



   With shielding enabled, queries from other POPs appear as incoming requests to the shield. If the shield doesn't know about the modified hostname, it doesn't know which service to match the request to. Including the origin's hostname in the domain list eliminates this concern.

10. Click the **Activate** button to deploy your configuration changes.

Caveats of shielding

Shielding not only impacts traffic and hit ratios, it affects configuration and performance. When you configure shielding, be aware of the following caveats.

Traffic and hit ratio caveats

Inbound traffic to a shield will be billed as regular traffic, including requests to populate remote POPs. Enabling shielding will incur some additional Fastly bandwidth charges, but will be offset by savings of your origin bandwidth (and origin server load). Pass-through requests will not go directly to the origin, they will go through the shield first.

Global HIT ratio calculation may seem lower than the actual numbers. Shielding is not taken into account when calculating the global hit ratio. If an edge node doesn't have an object in its cache, it reports a miss. Local MISS/Shield HIT gets reported as a miss and a hit in the statistics, even though there is no call to the backend. It will also result in one request from the edge node to the shield. Local MISS/Shield MISS will result in two requests, because we will subsequently fetch the resource from your origin. For more information about caching with shielding see our article Understanding Cache HIT and MISS with Shielding Services (/guides/performance-tuning/understanding-cache-hit-and-miss-headers-with-shielded-services).

Configuration caveats

You will be unable to manually define backends using VCL. Shielding at this level is completely dependent on backends being defined as actual objects through the web interface or API. Other custom VCL (/guides/vcl/uploading-custom-vcl) will work just fine.

You can only set one shield total if automatic load balancing is selected via the web interface. If you've selected auto load balancing, you can only select one shield total. You must use custom VCL to use multiple shields when auto load balancing is set.

Enabling sticky load balancing and shielding at the same time requires custom VCL. Sticky load balancers use `client.identity` to choose where to send the session. The `client.identity` defaults to the IP request header. That's fine under normal circumstances, but if you enable shielding, the IP will be the original POP's IP, not the client's IP. Thus, to enable shielding and a custom sticky load balancer, you want to use the following:

```
if (req.http.fastly-ff) {
  set client.identity = req.http.Fastly-Client-IP;
}
```

You'll need to use caution when changing the host header before it reaches the shield. Fastly matches a request with a host header. If the host header doesn't match to a domain within the service an error of 500 is expected. Also, purging conflicts can occur if the host header is changed to a domain that exists in a different service.

For example, say Service A has hostname `a.example.com` and Service B has hostname `b.example.com`. If Service B changes the host header to `a.example.com`, then the edge will think the request is for Service B but the shield will think the request is for Service A.

When you purge an object from Service B and not from Service A, the shield will serve the old object that you wanted to purge to the edge, since the purge went out to Service B and not Service A. You will want to purge the object from both Service A and Service B. However, this opens the door for confusion and error.

VCL gets executed twice: once on the shield node and again on the edge node. Changes to `beresp` and `resp` can affect the caching of a URL on the shield and edge. Consider the following examples.

Say you want Fastly to cache an object for one hour (3600 seconds) and then ten seconds on the browser. The origin sends `Cache-Control: max-age=3600`. You unset `beresp.http.Cache-Control` and then reset `Cache-Control` to `max-age=10`. With shielding enabled, however, the result will not be what you expect. The object will have `max-age=3600` on the shield and reach the edge with `max-age=10`.

A better option in this instance would be to use `Surrogate-Control` and `Cache-Control` response headers. `Surrogate-Control` overrides `Cache-Control` and is stripped after the edge node. The `max-age` from `Cache-Control` will then communicate with the browser. The origin response headers would look like this:

```
Surrogate-Control: max-age=3600
Cache-Control: max-age=10
```

Another common pitfall involves sending the wrong `Vary` header to an edge POP. For example, there's VCL that takes a specific value from a cookie, puts it in a header, and that header is then added to the `Vary` header. To maximize compatibility with any caches outside of your control (such as with shared proxies as commonly seen in large enterprises), the `Vary` header is updated in `vcl_deliver`, replacing the custom header with `Cookie`. The code might look like this:

```
vcl_recv {
  # Set the custom header
  if (req.http.Cookie ~ "ABtesting=B") {
    set req.http.X-ABtesting = "B";
  } else {
    set req.http.X-ABtesting = "A";
  }
  ...
}

...

sub vcl_fetch {
  # Vary on the custom header
  if (beresp.http.Vary) {
    set beresp.http.Vary = beresp.http.Vary ", X-ABtesting";
  } else {
    set beresp.http.Vary = "X-ABtesting";
  }
  ...
}

...

sub vcl_deliver {
  # Hide the existence of the header from downstream
  if (resp.http.Vary) {
    set resp.http.Vary = regsub(resp.http.Vary, "X-ABtesting", "Cookie");
  }
}
```

When combined with shielding, however, the effect of the above code will be that edge POPs will have `Cookie` in the `Vary` header, and thus will have a terrible hit rate. To work around this, amend the above VCL so that `Vary` is only updated with `Cookie` when the request is not coming from another Fastly cache. The `Fastly-FF` header is a good way to tell. The code would look something like this (including the same `vcl_recv` from the above example):

```
# Same vcl_recv from above code example

sub vcl_fetch {
  # Vary on the custom header, don't add if shield POP already added
  if (beresp.http.Vary !~ "X-ABtesting") {
    if (beresp.http.Vary) {
      set beresp.http.Vary = beresp.http.Vary ", X-ABtesting";
    } else {
      set beresp.http.Vary = "X-ABtesting";
    }
  }
  ...
}

...

sub vcl_deliver {
  # Hide the existence of the header from downstream
  if (resp.http.Vary && !req.http.Fastly-FF) {
    set resp.http.Vary = regsub(resp.http.Vary, "X-ABtesting", "Cookie");
  }
}
```

§ Tracking your origin's name, IP, and port (/guides/performance-tuning/tracking-your-origins-name-ip-and-port)

Fastly provides three values captured in `vcl_fetch` that allow you to see and track origin information:

- `beresp.backend.name`
- `beresp.backend.port`
- `beresp.backend.ip`

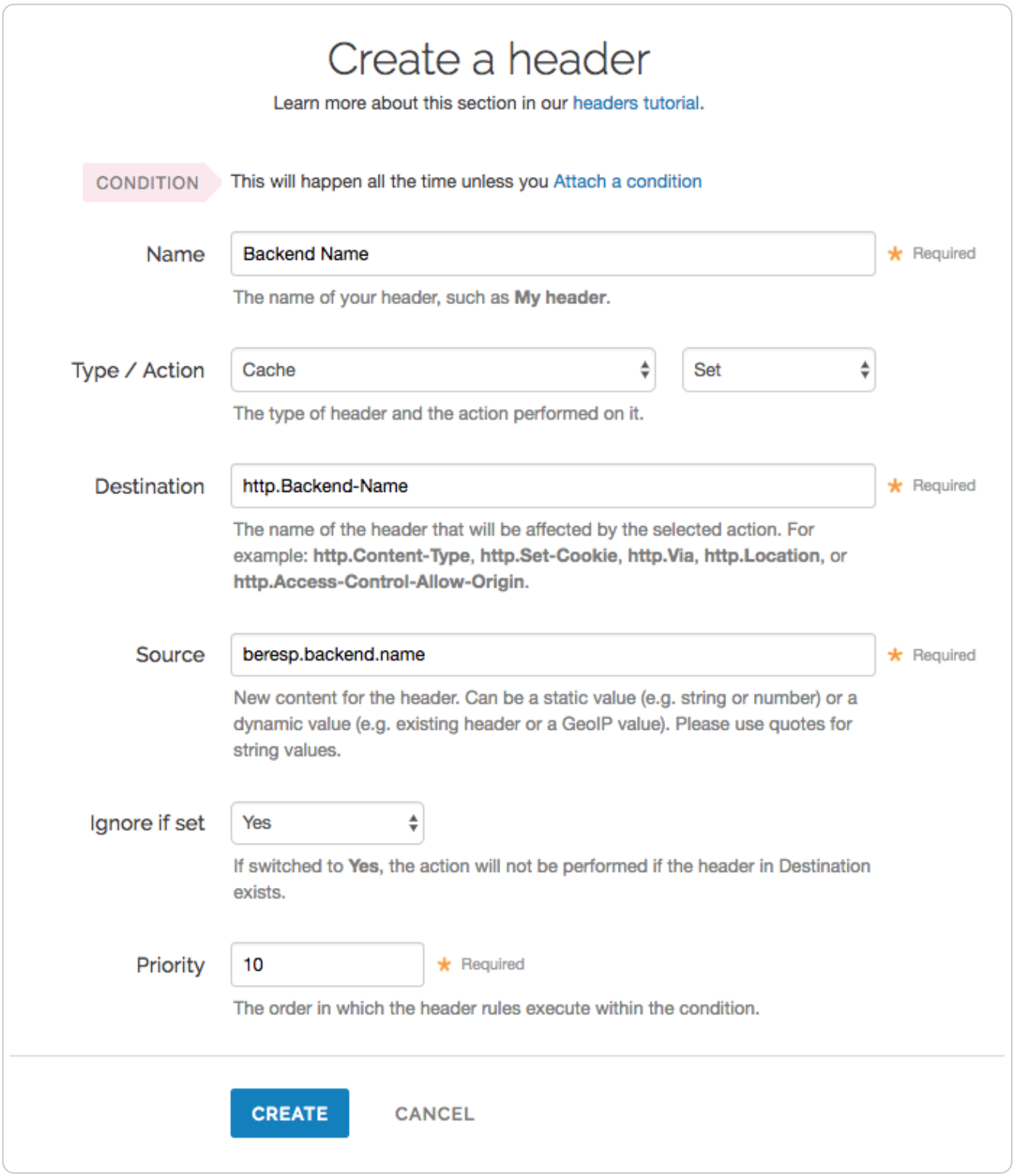
While these three values are immensely useful, you may want to use this information within `vcl_deliver` for things like response information or remote log streaming. You can do this by:

1. Creating cache headers that capture the origin information.
2. Adding a response header to the log format to capture the response output.

Capturing the Origin Information

To track your origin's name, IP, and port, you need to create two separate headers: one that captures the origin name and another that captures the origin's IP and port (e.g., 80, 443).

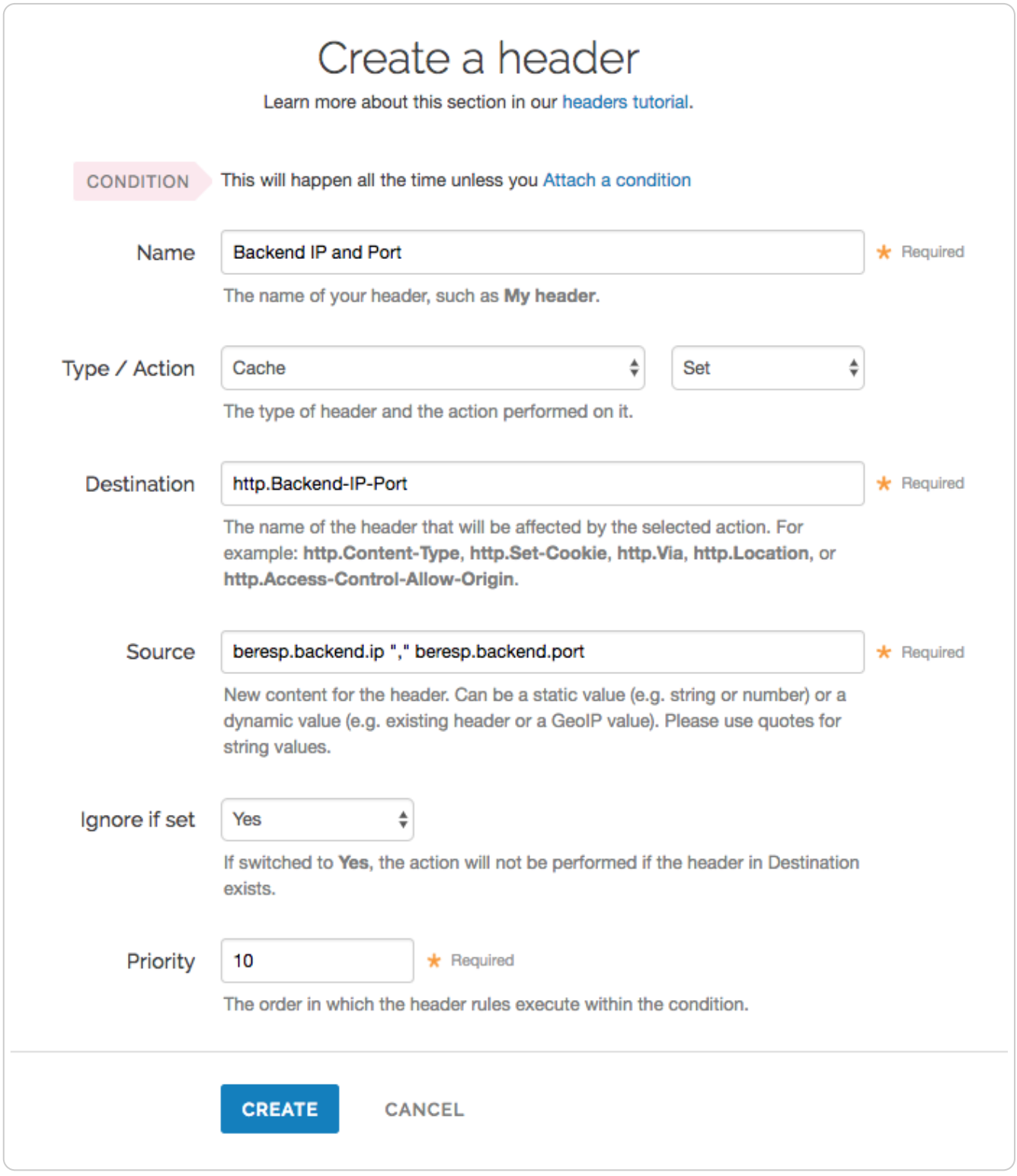
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button to create a header that captures the name of your origin. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Backend Name` (or preferred name). This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Backend-Name` (or preferred header variable).
   - In the **Source** field, type `beresp.backend.name`.

- From the **Ignore if set** menu, select **Yes**.
- In the **Priority** field, type `10`.

7. Click the **Create** button. The new header appears on the **Content** page.
8. Click the **Create header** button to create a header that captures the IP and port of your origin. The Create a header page appears.



9. Fill out the **Create a header** fields as follows:

- In the **Name** field, type `Backend IP and Port` (or preferred name). This name is displayed in the Fastly interface.
- From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
- In the **Destination** field, type `http.Backend-IP-Port` (or preferred header variable).
- In the **Source** field, type `beresp.backend.ip "," beresp.backend.port`.
- From the **Ignore if set** menu, select **Yes**.
- In the **Priority** field, type `10`.

10. Click the **Create** button. The new header appears on the **Content** page.
11. Click the **Activate** button to deploy your configuration changes.

Adding a Response Header to the Log Format

The values captured in a header within `vcl_fetch` will flow to `vcl_deliver`. For example, there will exist a `resp.http.Backend-Name` header in `vcl_deliver` that corresponds to `beresp.http.Backend-Name` in `vcl_fetch`. By default, the response header will be included in the response output.

Unfortunately, with remote log streaming, you cannot add the `vcl_fetch` header, `beresp.http.Header-Name`, to the log format. However, you can add its cousin in `vcl_deliver`, `resp.http.Header-Name`.

Add `resp.http.Header-Name` to the log format that you configured by following the instructions in the Remote Log Streaming guide (/guides/streaming-logs/setting-up-remote-log-streaming). Using the example above, you would add `resp.http.Backend-Name` and `resp.http.Backend-IP-Port`.

Important Notes

Regarding Shielding

Notice within the example the field **Ignore if set** is set to **Yes**. With shielding, the VCL is executed twice, once on the shield and again on the edge node. This setting will display the original information from the origin without overriding it on the edge node.

You can also set the field **Ignore if set** to **No** with shielding enabled. In this scenario, the edge node captures the shield's information within `beresp.backend.name`, `beresp.backend.ip`, and `beresp.backend.port`.

If remote log streaming is configured, remember it is executed twice. Thus the first log (from the shield node) will have the origin's information and the second log (from the edge node) will have the shield's information.

Regarding Security

For security purposes, you may want to track the information in logging but not display all or some of it in the response. This is possible but requires custom VCL to strip the information after sending the log line from the edge node.

> ❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Don't forget to read our guide to using custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl) before you begin. Remember to include the entire boilerplate if you do not intend to override the Fastly default settings.

Then add the following snippet within `vcl_deliver` with the headers you want to strip. Using our example above, we continue to send the value of the origin's name within the response. We strip the origin's IP and port from the response information.

```
sub vcl_deliver {
#FASTLY deliver

  if (!req.http.Fastly-FF) {
    unset resp.http.Backend-IP-Port;
  }

  return(deliver);
}
```

§ Understanding cache HIT and MISS headers with shielded services (/guides/performance-tuning/understanding-cache-hit-and-miss-headers-with-shielded-services)

Here's some help deciphering cache hit and miss headers when you have shielding enabled (/guides/performance-tuning/shielding). Let's look at the following requests for the same object if you had run a cURL command in your terminal (for example, `curl -svo /dev/null www.example.com`) to return the Fastly headers.

The first request for an object using the above cURL command might produce output something like this:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3120-SJC
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
```

For this first request, the two cache-nodes in X-Served-By show that shielding is turned on, with `cache-iad2120-IAD` serving as the delivering cache node at the shield datacenter and `cache-sjc3120-SJC` serving as the delivering cache node at the "local" datacenter. The `X-Cache: MISS, MISS` indicates that the requested object was neither in the shield cache (a MISS) nor the local delivering node (also a MISS). The `X-Cache-Hits` reflects that same MISS information because it displays `0, 0`.

The second request for an object using the above cURL command might produce output something like this:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3120-SJC
X-Cache: MISS, HIT
X-Cache-Hits: 0, 1
```

This second time, we hit the same local cache-node (`cache-sjc3120-SJC`) and got a HIT. The MISS listed for `cache-iad2120-IAD` reflects the state of that node the last time it was queried for that object and not its current state, which at the time of the first request, was a MISS. The object is now cached in both datacenters.

Waiting a minute or two and requesting the same object a third time using the above cURL command might produce output something like this:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3122-SJC
X-Cache: MISS, HIT
X-Cache-Hits: 0, 1
```

This third request shows a new cache (`cache-sjc3122-SJC`) being selected from the local datacenter. It registers as a HIT as the object is cached in the local datacenter, with the MISS still reflecting the state of the shield datacenter when it was originally requested. The `X-Cache-Hits` shows `0, 1` reflecting the `0` from the shield datacenter and the `1` for the first hit on cache-sjc-3122-SJC.

Keep in mind that if the closest delivering cache node exists in the shield datacenter, you will only see a single server and HIT data such as:

```
X-Served-By: cache-iad2120-IAD
X-Cache: HIT
X-Cache-Hits: 1
```

After a purge of the object, requesting the object again via the above cURL command will produce results similar to the first request scenario. For example:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3120-SJC
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
```

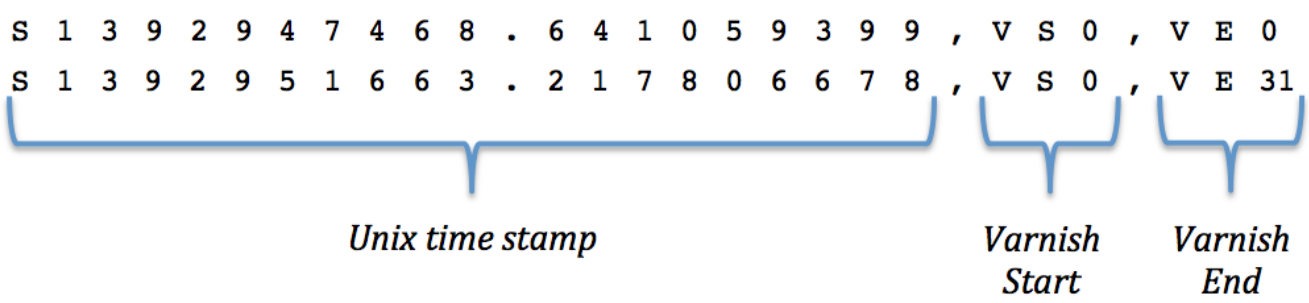
§ Understanding the X-Timer header (/guides/performance-tuning/understanding-the-xtimer-header)

If you look at the raw headers returned with a response from a Fastly cached asset, you will notice some extra headers tacked on. One in particular is X-Timer. This header provides timing information about the journey of a request from end to end.

Here are two examples of X-Timer headers:

- `S1392947468.641059399,VS0,VE0` (a cache HIT)
- `S1392951663.217806578,VS0,VE31` (a cache MISS)

Let's break these headers down into their parts, separated by commas, and examine what each part means.



The first section of the header, starting with `S`, represents a Unix timestamp (https://en.wikipedia.org/wiki/Unix_time) of the start of the request on our edges.

The next section, `VS` or "varnish start," represents the start of the varnish part of the request's journey. This should always be 0 (we've got to start counting somewhere).

And the last section, `VE` or "varnish end," represents the sum of the length of the trip. For cache HITs, the length of the trip will nearly always be 0 (not actually zero, but less than a millisecond is rounded down). For cache MISSs, the number represents the number of milliseconds it took to

retrieve the data from your origin server and send the response back to the requester. In the example above, it took 31ms to retrieve the data.

★ **TIP:** Interested in functions and variables that allow you to control dates and times using custom VCL (/guides/vcl/guide-to-vcl)? We have a guide (/guides/vcl/date-and-time-related-vcl-features) describing which ones we support.

---

§ Using edge side includes (ESI) (/guides/performance-tuning/using-edge-side-includes)

You can implement basic edge side includes (http://www.w3.org/TR/esi-lang) (ESI) through Fastly using custom VCL (/guides/vcl/guide-to-vcl). Fastly supports the following ESI elements:

- include
- comment
- remove

We don't support the following ESI language elements:

- inline
- choose | when | otherwise
- try | attempt | except
- vars
- ESI Variables

• Guides (/guides/) > Security > Access Control Lists (/guides/access-control-lists/)

§ About ACLs (/guides/access-control-lists/about-acls)

Malicious actors can present themselves in a variety of ways on the internet. Automated tools can scrape information from your website, bots can probe your application for vulnerabilities, and hackers can exploit them. If you detect threats like these, you may want to use an ACL to prevent the offending IP addresses from ever accessing your information resources again. An ACL is a list of permissions that Varnish (/guides/vcl/guide-to-vcl) uses to grant or restrict access to URLs within your services (/guides/basic-setup/working-with-services).

Ways of creating ACLs

There are two ways of creating ACLs for your Fastly services. You can use Edge ACLs to programmatically create and manipulate ACLs with the Fastly API, or you can manually create the ACL in VCL and upload it:

- **Edge ACLs:** You can use Edge ACLs (/guides/access-control-lists/about-edge-acls) to attach an ACL to a service with versionless ACL entries that are stored separately from your VCL configuration. You can use the Fastly API to programmatically add, remove, and update ACLs and their entries. We recommend this option if you want to integrate your website or application with an ACL.
- **Uploading custom VCL:** You can manually create an ACL (/guides/access-control-lists/manually-creating-access-control-lists) in VCL and upload it (/guides/vcl/uploading-custom-vcl). We recommend this option if you have simple access control requirements and can hardcode a few IP addresses in your VCL. ACLs that are manually created are versioned with your services, and any changes to the ACL will require changes to your VCL.

---

§ About Edge ACLs (/guides/access-control-lists/about-edge-acls)

Edge ACLs allow you to attach an access control list (ACL) to a service with versionless ACL entries that are stored separately from your VCL configuration. You can use the Fastly API to programmatically add, remove, and update ACLs and their entries.

How Edge ACLs work

Like Edge Dictionaries (/guides/edge-dictionaries/about-edge-dictionaries), Edge ACLs have two major components: The ACL itself (/guides/access-control-lists/creating-and-using-edge-acls), and the ACL entries (/guides/access-control-lists/creating-and-manipulating-edge-acl-entries) within it. ACLs are containers for ACL entries, which store IP addresses that can be used to whitelist or blacklist access to resources. Every ACL is attached to a version of a service, but ACL entries are versionless and any changes will become effective immediately.

When ACLs might be useful

- E-commerce companies preventing scraping from certain IP ranges
- Offices restricting access to their administrative portals
- Advertising technology companies blocking bad-actors at edge
- Mobile applications accepting only calls from specific proxies or IP ranges
- System administrators restricting access to groups of backends from an office IP address or subnet range

Getting started

To create an ACL and use it within your service, you'll need to perform the following steps:

1. Create an empty ACL (/guides/access-control-lists/creating-and-using-edge-acls) in a working version of a service that's unlocked and not yet activated.
2. Activate the new version of the service (/guides/basic-setup/working-with-services#editing-and-activating-versions-of-services) you associated with the empty ACL.
3. Add ACL entries (/guides/access-control-lists/creating-and-manipulating-edge-acl-entries) to the newly created ACL.

Once the ACL is created, properly associated, and filled with ACL entries, it can be called in your service.

Putting Edge ACLs to work

After you've used the Fastly API to create an ACL and add ACL entries, the VCL for the ACLs and ACL entries will be automatically generated, as shown below. In this example, the name of the ACL is `office_ip_ranges`.

```
# This VCL is automatically generated when you add an Edge ACL and entries with the Fastly API
# In this example, the ACL name is "office_ip_ranges"
acl office_ip_ranges {
  "10.1.1.0"/16;                                # internal office
  "8.19.222.194";                               # remote VPN office
  "FE80:0000:0000:0000:0202:B3FF:FE1E:8329";    # ipv6 address remote
}
```

You can upload custom VCL (/guides/vcl/uploading-custom-vcl) to add logic to interact with ACLs. In this example, the `office_ip_ranges` ACL is used as a whitelist. All access to `/admin` is denied by default, but IP addresses in the `office_ip_ranges` ACL are allowed to access `/admin` without restriction.

```
sub vcl_recv {
  # block all requests to Admin pages from IP addresses not in office_ip_ranges
  if (req.url ~ "^/admin" && ! (client.ip ~ office_ip_ranges)) {
    error 403 "Forbidden";
  }
}
```

ACL entries have a boolean option for negation which allows you to specify whether or not the IP address is whitelisted or blacklisted. You can set the option to true (1) to blacklist (deny), or false (0) to whitelist (allow). We *do not* recommend mixing both negated and non-negated entries in the same ACL. Doing so might have the opposite effect depending on the condition you specify in the VCL.

Limitations

When creating Edge ACLs (/guides/access-control-lists/creating-and-using-edge-acls), keep the following limitations in mind as you develop your service configurations:

- **ACLs created with custom VCL cannot be manipulated using the API.** If you create an ACL using the API, you can use the API to make changes to it, and use custom VCL to interact with it. If you create an ACL using custom VCL (/guides/access-control-lists/manually-creating-access-control-lists), that ACL must always be manipulated via custom VCL.
- **Custom VCL manipulations are always versioned.** This is true for both ACLs created using custom VCL and for any logic created to interact with ACLs.
- **ACLs are limited to 1000 ACL entries.** If you find your ACLs approaching this limit, you may want to review this with your account manager. We may be able to help you figure out an even more efficient way to do things with your ACLs.
- **Event logs don't exist for ACL changes.** If you use the API to add, update, or remove an ACL entry, there will be no record of it. The only record of a change will exist when you compare service versions (/guides/basic-setup/working-with-services#comparing-different-service-versions) to view the point at which the ACL was associated with the service version in the first place.
- **When you delete an ACL, you'll only delete it from the service version you're editing.** ACLs are tied to versions and can be cloned and reverted. When using ACLs, we want you to be able to do things like delete an ACL from a current version of your service in order to roll back your configuration to a previous version using as few steps as possible.
- **When you delete an ACL, we remove the ACL entries inside it.** When you delete an ACL we remove the entries that live inside that container, but they are not deleted from the ACL in earlier versions of your configuration. You can roll back to a previous version of a service to restore the ACL and the ACL entries.

- **ACL entry deletions are permanent.** If you delete an ACL entry, the entry is permanently removed from all service versions and cannot be recovered.

§ Creating and manipulating Edge ACL entries (/guides/access-control-lists/creating-and-manipulating-edge-acl-entries)

Edge ACL (/guides/access-control-lists/about-edge-acls) entries contain IP addresses that can be used to whitelist or blacklist access to resources. ACL entries are versionless, and any changes will become effective immediately, regardless of version.

Parameters

Edge ACL entries have the following parameters:

- **service_id:** The ID of the Fastly service the ACL is associated with.
- **acl_id:** The ID of the ACL.
- **id:** The ID of the ACL entry.
- **ip:** The IP address contained within the ACL entry.
- **subnet:** Optional. The range of IP addresses within a single ACL entry.
- **negated:** Defaults to false. A number indicating false (`0`) to allow, or true (`1`) to negate. Note that we *do not* recommend mixing both negated and non-negated entries in the same ACL.
- **comment:** Optional. A descriptive comment indicating why you created the ACL entry.

Creating an ACL entry

To add an entry to an existing ACL, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/acl/<acl_id>/entry -d 'ip=127.0.0.1&subnet=16&negated=0&comment=test'
```

The response will look like this:

```
{
    "acl_id": "<acl_id>",
    "comment": "test",
    "created_at": "2016-04-22T19:14:02+00:00",
    "deleted_at": null,
    "id": "<acl_entry_id>",
    "ip": "127.0.0.1",
    "negated": "0",
    "service_id": "<service_id>",
    "subnet": 16,
    "updated_at": "2016-04-22T19:14:02+00:00"
}
```

Viewing ACL entries

To see information related to a single ACL entry, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/<service_id>/acl/<acl_id>/entry/<acl_entry_id>
```

The response will look like this:

```
{
    "acl_id": "<acl_id>",
    "comment": "",
    "created_at": "2016-04-22T19:18:42+00:00",
    "deleted_at": null,
    "id": "<acl_entry_id>",
    "ip": "127.0.0.5",
    "negated": "0",
    "service_id": "<service_id>",
    "subnet": 16,
    "updated_at": "2016-04-22T19:18:42+00:00"
}
```

To view a list of all ACL entries attached to a particular ACL, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/acl/<acl_id>/entries
```

The response will look like this:

```
[
    {
        "acl_id": "<acl_id>",
        "comment": "",
        "created_at": "2016-04-22T19:13:03+00:00",
        "deleted_at": null,
        "id": "<acl_entry_1_id>",
        "ip": "127.0.0.1",
        "negated": "0",
        "service_id": "<service_id>",
        "subnet": 16,
        "updated_at": "2016-04-22T19:13:03+00:00"
    },
    {
        "acl_id": "<acl_id>",
        "comment": "",
        "created_at": "2016-04-22T19:14:02+00:00",
        "deleted_at": null,
        "id": "<acl_entry_2_id>",
        "ip": "127.0.0.2",
        "negated": "0",
        "service_id": "<service_id>",
        "subnet": 16,
        "updated_at": "2016-04-22T19:14:02+00:00"
    }
]
```


Updating ACL entries

There are two ways to update ACL entries: you can update a single ACL entry, or you can update multiple ACL entries at the same time.

Updating a single ACL entry

To update an existing ACL entry, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -X PATCH https://api.fastly.com/service/<service_id>/acl/<acl_id>/entry/<acl_entry_id> -d 'ip=127.0.0.2&subnet=32&negated=0&comment=allow'
```

The response will look like this:

```
{
    "acl_id": "<acl_id>",
    "comment": "allow",
    "created_at": "2016-04-22T19:18:42+00:00",
    "deleted_at": null,
    "id": "<acl_entry_id>",
    "ip": "127.0.0.2",
    "negated": "0",
    "service_id": "<service_id>",
    "subnet": 32,
    "updated_at": "2016-04-22T19:18:42+00:00"
}
```

Updating multiple ACL entries

You can also update multiple ACL entries at the same time. Include an `entries` array of changes in the API call and pass an operation (`op`) parameter for every change. Possible `op` values are `create`, `update`, and `delete`.

To update multiple ACL entries at the same time, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -H "Content-type: application/json" -X PATCH https://ap
i.fastly.com/service/<service_id>/acl/<acl_id>/entries -d '{"entries":[{"op": "create", "ip":
"192.168.0.1","subnet": "8"},{"op": "update", "id": "<acl_entry_id>", "ip": "192.168.0.2", "sub
net": "16"},{"op": "delete", "id": "<acl_entry_id>"}]}'
```

The response will look like this:

```
{
  "status":"ok"
}
```

Deleting an ACL entry

> ⚠ **WARNING:** ACL entry deletions are permanent. If you delete an ACL entry, the entry is permanently removed from all service versions and cannot be recovered.

To permanently delete an ACL entry, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id>/ac
l/<acl_id>/entry/<acl_entry_id>
```

The response will look like this:

```
{
  "status":"ok"
}
```

§ Creating and using Edge ACLs (/guides/access-control-lists/creating-and-using-edge-acls)

Edge ACLs (/guides/access-control-lists/about-edge-acls) are lists of permissions that Varnish uses to grant or restrict access to URLs within your services. You can use the Fastly API to programmatically add, remove, and update Edge ACLs and their entries. To create an Edge ACL and use it within your service, you'll need to perform the following steps:

1. Create an empty ACL (/guides/access-control-lists/creating-and-using-edge-acls) in a working version of a service that's unlocked and not yet activated.
2. Activate the new version of the service (/guides/basic-setup/working-with-services#editing-and-activating-versions-of-services) you associated with the empty ACL.
3. Add ACL entries (/guides/access-control-lists/creating-and-manipulating-edge-acl-entries) to the newly created ACL.

Once the ACL is created, properly associated, and filled with ACL entries, it can be called in your service.

Attributes

Edge ACLs have the following attributes:

- **Service ID:** The ID of the Fastly service the ACL is associated with.
- **VCL Version Number:** The VCL version number the ACL is associated with. Note that the ACL will continue to reside within subsequently cloned counterparts.
- **ACL Name:** The name of the ACL.
- **ACL ID:** The unique identifier of the ACL.

Creating an ACL

To start using an ACL, you'll need to create an empty one within a version of a service that's unlocked and not yet activated. Make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/vers
ion/<vcl_version_number>/acl -d name=my_acl
```

The response will look like this:

```
{
    "id": "<vcl_version_number>",
    "name": "my_acl",
    "service_id": "<service_id>",
    "version": "1",
    "created_at": "2016-04-14 21:23:21",
    "updated_at": "2016-04-14 21:23:21"
}
```

Be sure to activate the new version of the service you associated with the empty ACL.

Viewing ACLs

To see information related to a single ACL (in this example, `my_acl`) attached to a particular version of a service, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/<vcl
_version_number>/acl/my_acl
```

The response will look like this:

```
{
    "id": "<acl_id>",
    "name": "my_acl",
    "service_id": "<service_id>",
    "version": "<vcl_version_number>",
    "created_at": "2016-04-14 21:23:21",
    "updated_at": "2016-04-14 21:23:21"
}
```

To view a list of all ACLs attached to a particular version of a service, make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/<vcl
_version_number>/acl
```

The response will look like this:

```
[
    {
        "id": "<acl_1_id>",
        "name": "my_new_acl",
        "service_id": "<service_id>",
        "version": "<vcl_version_number>",
        "created_at": "2016-04-14 21:23:21",
        "updated_at": "2016-04-15 17:23:09"
},
    {
        "id": "<acl_2_id>",
        "name": "my_other_acl",
        "service_id": "<service_id>",
        "version": "<vcl_version_number>",
        "created_at": "2016-04-14 21:23:21",
        "updated_at": "2016-04-15 17:23:09"
    }
]
```

Deleting an ACL

Deleting an ACL deletes the ACL and all of its associated entries. To delete an ACL (in this example, `my_new_acl`), make the following API call in a terminal application:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id>/ve
rsion/<vcl_version_number>/acl/my_new_acl
```

The response will look like this:

```
{
  "status":"ok"
}
```

§ Manually creating access control lists (/guides/access-control-lists/manually-creating-access-control-lists)

Varnish (/guides/vcl/guide-to-vcl) allows you to use Access Control Lists (ACLs), a feature that enables fast matching of a client's IP address against a list of defined IP addresses. An ACL looks like this:

```
# Who is allowed access ...
acl local {
  "localhost";
  "192.168.1.0"/24; /* and everyone on the local network */
  ! "192.168.1.23"; /* except for the dial-in router */
}
```

Defining an ACL

Using ACLs requires you to create and add custom VCL to Fastly's boilerplate VCL. To define an ACL in your Fastly configuration:

1. Ask support (mailto:support@fastly.com) to enable custom VCL uploading (/guides/vcl/uploading-custom-vcl) for your account.
2. Read about how to mix and match custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl) with Fastly's VCL.
3. Create a custom VCL file with your ACL definitions included in the appropriate location. Use the example shown below as a guide. You can reference the ACL in your configuration (`vcl_recv`) using a match operation which can be located above or below `#FASTLY recv` - the placement only matters for the order of operations within Varnish's execution of your configuration.

   ```
   # If you are using the "include" keyword
   include "myACL1.vcl";

   # And/or if you are using an actual ACL block
   acl local {
     "localhost";
     "192.168.1.0"/24; /* and everyone on the local network */
     ! "192.168.1.23"; /* except for the dial-in router */
   }

   sub vcl_recv {
     # block any requests to Admin pages not from local IPs
     if (req.url ~ "^/admin" && client.ip !~ local) {
       error 403 "Forbidden";
     }
   }
   ```

4. Upload the file (/guides/vcl/uploading-custom-vcl) in the Varnish Configuration area of your service.

Shielding Caveats

Be aware that if you've enabled shielding (/guides/performance-tuning/shielding#enabling-shielding), you need to ensure the client IP check is only executed at the edge. For example:

```
sub vcl_recv {
  # block any requests to Admin pages not from local IPs
  if (req.url ~ "^/admin" && client.ip !~ local && !req.http.Fastly-FF){
    error 403 "Forbidden";
  }
}
```

The `client.ip` provides the source address connecting to Fastly. In the case of Origin Shielding (/guides/detailed-product-descriptions/about-fastlys-full-site-delivery-features#origin-shielding), that address gets overwritten as one Fastly POP (https://www.fastly.com/network-map) connects to another. The `Fastly-FF` header in the above example ensures this code is not executed on the shield server because it gets added by the edge node.

You can create different behaviors based on any other attributes from a request as well, such as location and cookie presence.

• Guides (/guides/) > Security > Monitoring and testing (/guides/monitoring-and-testing/)

§ Monitoring account activity with event logs (/guides/monitoring-and-testing/monitoring-account-activity-with-event-logs)

Event logs keep track of events related to your services, account, and users. You can use event logs to determine which changes were made and by whom. Event logs are available for retrieval via the Fastly API. A limited subset of your services' event logs are also available within the web interface. Event log data is currently retained indefinitely.

Types of events that are logged

Event logs record many types of events that users can perform:

- **Version management:** version activation, version deactivation, and version lock
- **User behavior:** log in, log out, failed login, account locked, account unlocked, and password updated
- **Security:** two-factor authentication enabled, two-factor authentication disabled, API token created, and API token deleted

- **Account management:** new invitation, invitation accepted, user updated, user deleted, new company addition, and company settings updated
- **Billing:** pricing plan change and billing contact change
- **Purging:** purge all
- **Dynamic Servers:** pool creation, pool deletion, pool update, server creation, server deletion, and server update
- **Configuration settings (creation, deletion, and updates)**: domain, backend, header, health check, cache setting, request setting, logging endpoint, VCL file, gzip setting, and synthetic response

When event logs might be useful

- Seeing who activated the most recent version of your service
- Reviewing who logged in to your account via the web interface
- Learning which users have two-factor authentication enabled
- Viewing recent service configuration setting changes

Accessing event logs

There are two ways to access event logs: by using the Fastly API or by viewing the latest events in the web interface. The events displayed in the web interface are a subset of those available via the API.

Using the API

You can use the `/events` API endpoint to access your event logs. Events can be filtered by `user_id`, `service_id`, `customer_id`, and `event_type`. For example, you could make the following API call in a terminal application to view all recent events:

```
curl -g -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/events?filter[customer_id]=x4x
CwxxJxGCx123Rx5xTx&page[number]=1&page[size]=1
```

The response will look like this:

```
{
  "data": [
      {
          "attributes": {
              "admin": false,
              "created_at": "2016-06-06T20:05:10Z",
              "customer_id": "x4xCwxxJxGCx123Rx5xTx",
              "description": "Version 2 was activated",
              "event_type": "version.activate",
              "ip": "127.0.0.0",
              "metadata": {
                  "version_number": 2
              },
              "service_id": "SU1Z0isxPaozGVKXdv0eY",
              "user_id": "4Pp0BW3UkBEJhG3N0kovLP"
          },
          "id": "5IH1QmNSV1Qi7jXc4oIZlZ",
          "type": "event"
      }
  ],
  "links": {
    "last": "https://api.fastly.com/events?filter[customer_id]=x4xCwxxJxGCx123Rx5xTx&page[numbe
r]=1&page[size]=1"
  }
}
```

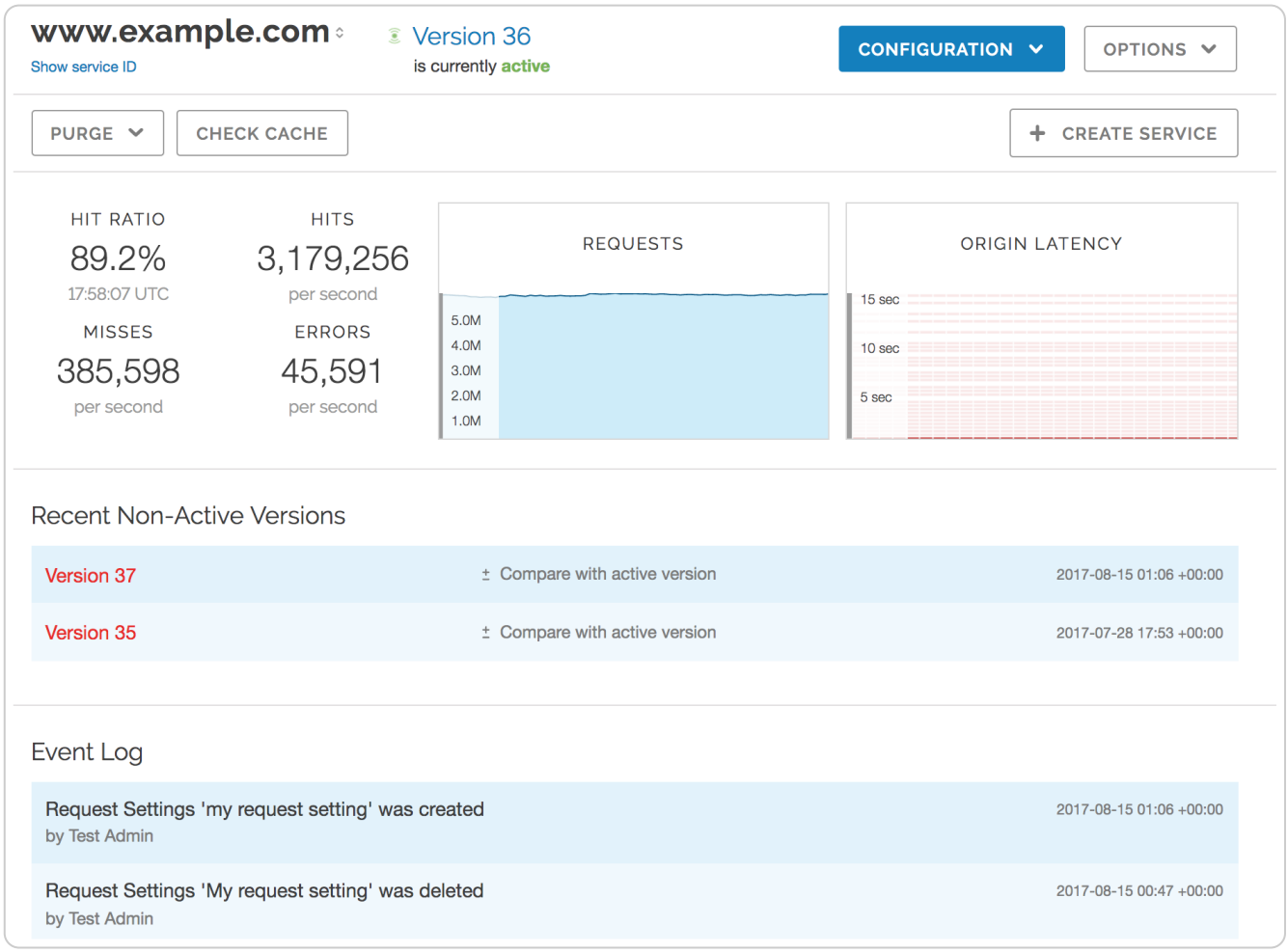

See the API documentation (/api/account#events) for more information.

Using the web interface

The web interface displays the last 20 service-related events for the selected service. Events related to users and accounts are not displayed in the web interface.

Follow these instructions to access the event logs for a service:

1. Log in to the Fastly web interface and click the **Configure** link.
2. The most recent service-related events are displayed near the bottom of the page, in the Event Log area.



§ Penetration testing your service behind Fastly (/guides/monitoring-and-testing/penetration-testing-your-service-behind-fastly)

We understand the need for our customers to validate the security of their service behind Fastly.

> **IMPORTANT:** Penetration tests that interfere with or disrupt the integrity or performance of Fastly services violate our acceptable use policy (https://www.fastly.com/acceptable-use). You must respond immediately to any communication from Fastly regarding your test to help ensure your testing does not adversely affect other customers or the Fastly network.

To perform security testing of your Fastly service configurations, create a Customer Support ticket by contacting Fastly via email at support@fastly.com (mailto:support@fastly.com) at least two (2) business days before you begin any security testing. In your ticket, include these details:

- the IDs of the services (/guides/account-management-and-security/finding-and-managing-your-account-info#finding-your-service-id) that will be tested
- the source IP address of the test
- the date of the test
- the start and end time of the test, including the time zone
- the contact information for the individual or third party performing the test, including a phone number and e-mail address
- whether or not the security test is likely to lead to significantly increased traffic volume

The following requirements apply to any security testing you perform:

- Only test Fastly services you own or are authorized by the owner to test. You may not perform tests against other customers without explicit permission or against Fastly-owned resources.
- Do not begin testing until after Fastly has responded affirmatively to your ticket and authorized your request.
- Update the ticket if either the scope or timeframe of your testing changes.
- If you discover vulnerabilities in the Fastly platform during your test, update the ticket with your findings as soon as possible so we can address them.

Fastly maintains programs for security (/guides/compliance/security-program) and technology compliance (/guides/compliance/technology-compliance). To perform an independent audit of these programs, contact sales@fastly.com (mailto:sales@fastly.com) to discuss purchase of Assurance Services (/guides/detailed-product-descriptions/assurance-services).

★ **TIP:** We welcome security professionals researching potential vulnerabilities in our network under our guidelines for reporting a security issue (https://www.fastly.com/security/report-security-issue).

---

§ Accessing Fastly's IP ranges (/guides/securing-

communications/accessing-fastlys-ip-ranges)

To help you whitelist Fastly's services through your firewall, we provide access to the list of Fastly's assigned IP ranges. You can access the list via URL:

https://api.fastly.com/public-ip-list (https://api.fastly.com/public-ip-list)

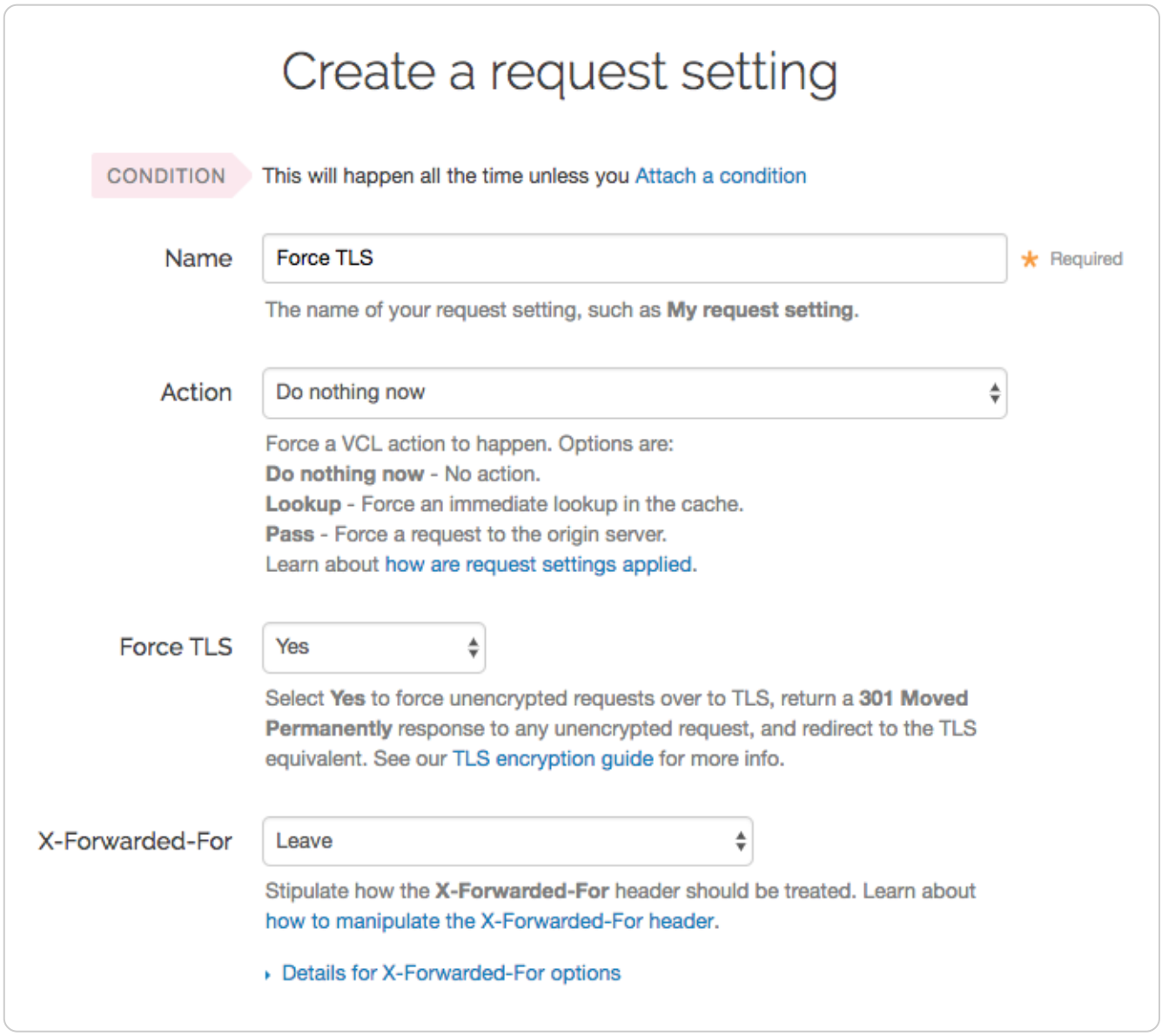
You can then automate the API call (for example, by running a script (https://github.com/jondade/IP-Whitelist-cron) as a cron job) to request the list of IPs to detect when the IP ranges change.

To make sure you have plenty of time to stay in sync, we post IP address announcements (https://status.fastly.com/incidents/sg8850gdhv26) along with other service announcements to our status page (https://status.fastly.com/), which you can subscribe (/guides/debugging/fastlys-network-status#subscribing-to-notifications) to.

§ Allowing only TLS connections to your site (/guides/securing-communications/allowing-only-tls-connections-to-your-site)

If you want to only allow TLS on your site, we have you covered. There is a switch built into the "Request Settings" that will allow you to force unencrypted requests over to TLS. It works by returning a **301 Moved Permanently** response to any unencrypted request, which redirects to the TLS equivalent. For instance, making a request for **http**://www.example.com/foo.jpeg would redirect to **https**://www.example.com/foo.jpeg.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Create request setting** button. The Create a request setting page appears.



6. Fill out the **Create a request setting** fields as follows:
   - In the **Name** field, type a human-readable name for the request setting. This name is displayed in the Fastly web interface.
   - From the **Force TLS** menu, select **Yes**.
7. Click the **Create** button to save your request setting changes.
8. Click the **Activate** button to deploy your configuration changes.

For more information about TLS-related issues, see our TLS guides (/guides/securing-communications/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

---

§ Domain validation for TLS certificates (/guides/securing-communications/domain-

validation-for-tls-certificates)

When you purchase our shared certificate (/guides/detailed-product-descriptions/tls-service-options#shared-tls-certificate-service) or shared wildcard certificate (/guides/detailed-product-descriptions/tls-service-options#shared-tls-wildcard-certificate-service) TLS options, our partner Certificate Authority (GlobalSign) must verify you control the domains requested and that you authorize us to request a certificate service on your behalf. You can choose:

- DNS text record verification (preferred)
- URL verification
- Email verification

Regardless of the verification method you use, be sure to follow our instructions (/guides/detailed-product-descriptions/tls-service-options) to begin the TLS ordering process.

DNS text record verification

We provide you with a unique DNS TXT record you need to add for the zone origin ("@"). The text of this entry will change depending on the certificate to which your domain is added. The meta tag will be formatted similar to one of the following (where the `{META TAG}` will change depending on the certificate):

- `@ IN TXT "globalsign-domain-verification={META TAG}"`
- `@ IN TXT "_globalsign-domain-verification={META TAG}"`

We will provide you with the appropriate text record listed above. Consult the documentation for your DNS server or hosted DNS provider for more information about how to add the record. This text record must be wholly separate from other text records. A prepended, inserted, or appended record will not work.

URL verification

We provide you with an HTML meta tag you need to add to a specifically named web page served at the requested domain or apex domain you're adding. Use the format `http:<REQUESTED APEX OR SUBDOMAIN>/.well-known/pki-validation/gsdv.txt` where `<REQUESTED APEX OR SUBDOMAIN>` is the domain being added to the certificate. The meta tag will be formatted similar to one of the following (where the `{META TAG}` text will change depending on the certificate):

- `<meta name="globalsign-domain-verification" content="{META TAG}" />`
- `<meta name="_globalsign-domain-verification" content="{META TAG}" />`

We will provide you with the appropriate meta tag listed above. This text must be served from the actual requested domain or root domain. For example, if you add the domain `www.example.com` to the certificate, GlobalSign will specifically query `http://www.example.com` or `http://example.com` during the verification process. The verification tag must be served from whatever resource is returned from that URL. GlobalSign will not follow redirects or request a file on that domain, such as `http://www.example.com/verify.html` or `http://www.example.com/index.html`.

Email verification

GlobalSign will give Fastly a list of acceptable email addresses to which they can send a validation email. Generally these email addresses match those that appear on the WHOIS record of the domain requested, plus the following:

- admin@domain.com
- administrator@domain.com
- hostmaster@domain.com
- postmaster@domain.com
- webmaster@domain.com

For entries requested for a subdomain, each of those addresses `@subdomain.domain.com` will also work (e.g., `admin@subdomain.domain.com`).

We will send you the list of acceptable email address. You will need to tell us which email address to use. GlobalSign will then send a verification email to the email address you specify. Once you receive the verification email, you will need to click on a link in that email and follow the instructions to complete the validation.

---

§ Enabling HSTS through Fastly (/guides/securing-communications/enabling-hsts-through-fastly)

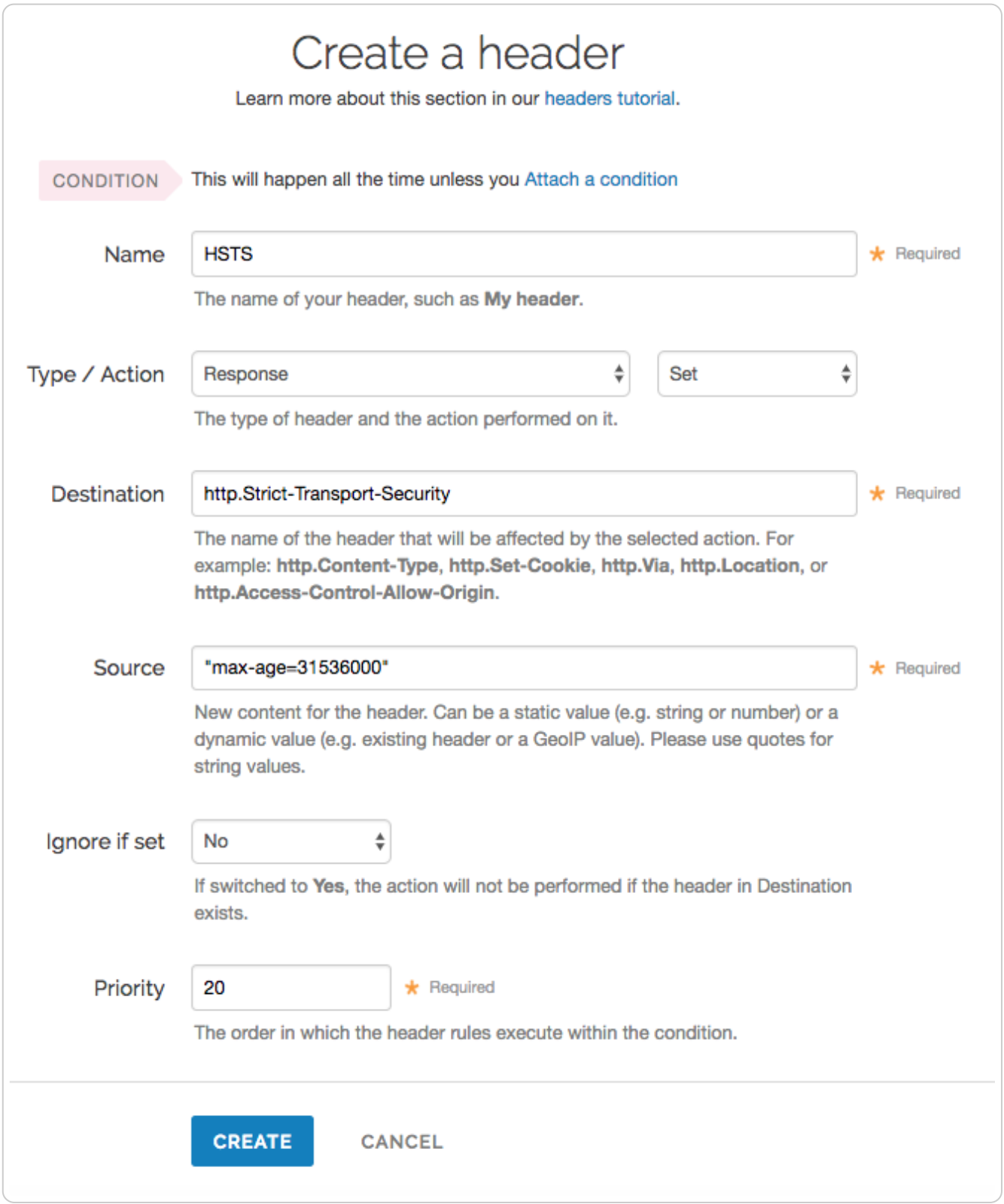
The HTTP Strict Transport Security (https://tools.ietf.org/html/rfc6797) (HSTS) security enhancement specification provides a way to force modern browsers to communicate only via the Transport Layer Security (TLS) protocol. Once enabled, it will force the browser to redirect (typically with a status code 307) to the HTTPS URL.

ℹ **NOTE:** HSTS only takes effect *after* a site has been visited on a trusted HTTPS connection. It doesn't replace the need to have redirects from your HTTP site.

Enabling HSTS

To enable HSTS, add a new header (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) as follows:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button to create a new header. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a human-readable name, such as `HSTS`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Response**, and from the **Action** menu select **Set**.
   - In the **Destination** field, type `http.Strict-Transport-Security`.

- In the **Source** field, type `"max-age=<max age in seconds>"`. For example, `"max-age=31536000"`. As described below, `max-age` is required and two additional HSTS options can be specified.
- Leave the **Ignore if set** menu and the **Priority** field set to their defaults (or set them as appropriate for your service).

7. Click the **Create** button.
8. Click the **Activate** button to deploy your configuration changes.

HSTS options

HSTS requires the max-age directive (https://tools.ietf.org/html/rfc6797#section-6.1.1) be set in order to function properly. It specifies how long in seconds to remember that the current domain should only be contacted over HTTPS. The example shown above sets `max-age` to one year (31536000 seconds = 1 year). You may want to experiment using a smaller value than what is shown.

Two additional options can be specified with the HSTS response header:

- `includeSubdomains` - This token applies HSTS to all of your site's subdomains. Before you include it, be certain none of your subdomains require functionality on HTTP in a browser. Ensure your TLS certificate is a wildcard or has coverage for all subdomain possibilities.

  > **IMPORTANT:** All subdomains will be unreachable on HTTP by browsers that have seen the HSTS header once `includeSubdomains` is enabled.

- `preload` - This token allows you to submit your domain for inclusion in a preloaded HSTS list that is built into several major browsers. Although the token is not part of the HSTS specification, including it in the header is a prerequisite for submitting to this preloaded list.

  > **WARNING:** Don't request browser preload inclusion unless you're sure that you can support HTTPS for the long term. Inclusion in the HSTS Preload List cannot be undone easily. See https://hstspreload.appspot.com/ (https://hstspreload.appspot.com/) for submission instructions and more information.

Combining all of these options together in the **Source** field would look like this:

`"Strict-Transport-Security: max-age=<max age in seconds>; includeSubDomains; preload"`

To disable HSTS for whatever reason, simply set the `max-age` to `0` on an HTTPS connection.

The HSTS Preload List is managed by a third party, not by Fastly. See https://hstspreload.appspot.com/ (https://hstspreload.appspot.com/) for more information.

Additional reading

- RFC 6797 (http://tools.ietf.org/html/rfc6797), which describes the HSTS specification
- the Wikipedia description (https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security) of HSTS, including the currently known limitations and a browser support list
- the OWASP.org explanation (https://www.owasp.org/index.php/HTTP_Strict_Transport_Security) of HSTS, including descriptions of the threats it addresses
- the Chromium Projects description (https://www.chromium.org/hsts) of HSTS and preloading HSTS sites

---

§ Setting up free TLS (/guides/securing-communications/setting-up-free-tls)

Customers can use our free shared domain TLS wildcard certificate to test TLS websites or applications using a Fastly URL (e.g., `https://<name>.global.ssl.fastly.net/`).

Before you begin

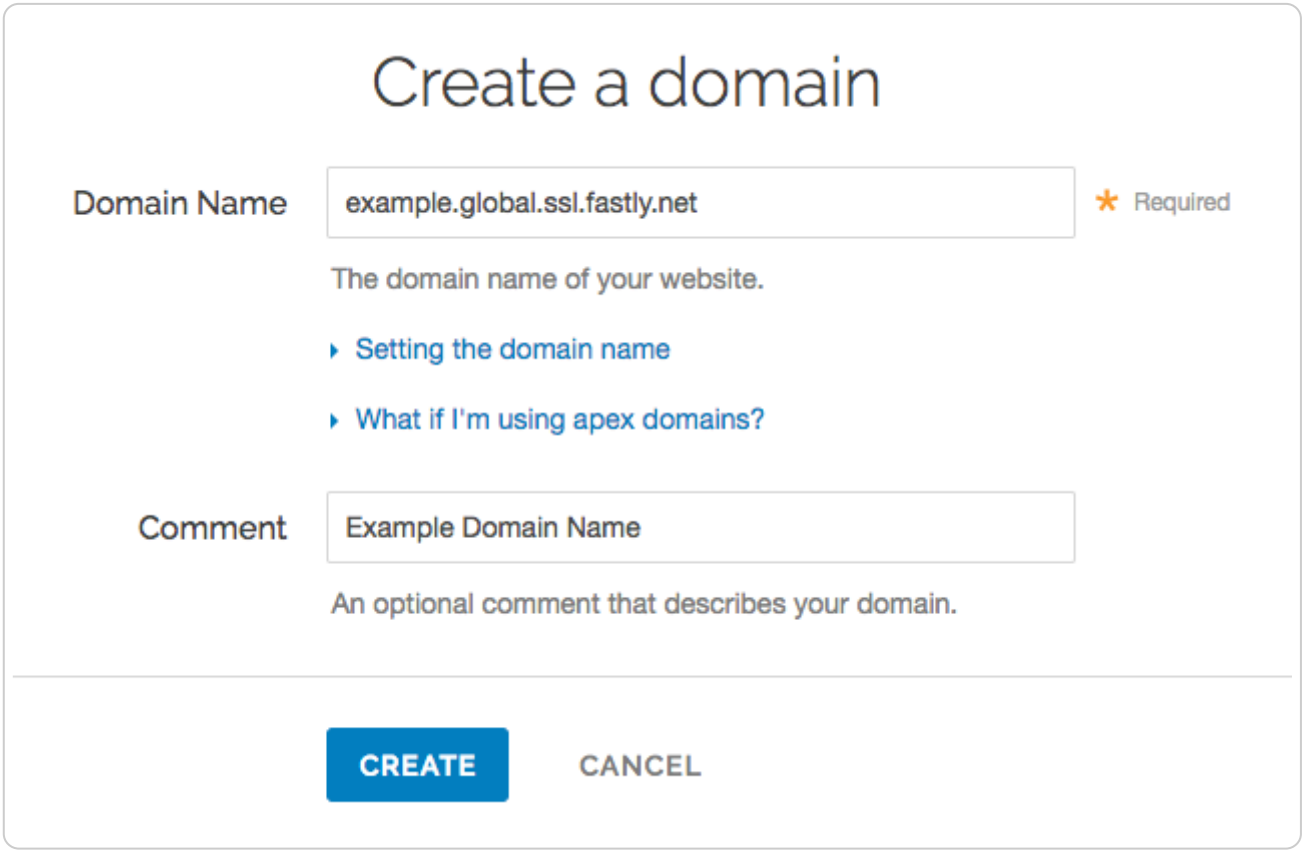
Before you begin setting up free TLS, understand the following:

- Free TLS uses a shared domain name and may not be suitable for a production environment if the domain name you use matters. For that, you'll need a paid TLS option (/guides/detailed-product-descriptions/tls-service-options).
- When using free TLS, you cannot DNS alias your own domain (`www.example.org`) to the shared domain (`example.global.ssl.fastly.net`). If you do, a TLS name mismatch warning will appear in the browser. The only way to avoid the mismatch error is to order a paid TLS option (/guides/detailed-product-descriptions/tls-service-options).
- Free TLS supports TLS protocols TLS 1.2 and higher.

Setting up free TLS for the first time

Follow the steps below to set up free TLS:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Create domain** button. The Create a domain page appears.



5. Fill out the **Create a domain** fields as follows:
   - In the **Domain Name** field, type `<name>.global.ssl.fastly.net`, where `<name>` is a single word that claims the domain you're creating. You cannot use a dot-separated name (e.g., `www.example.org.global.ssl.fastly.net`) because TLS certificates do not support nesting. If the name you choose has already been claimed, you will need to pick a different one.
   - In the **Comment** field, type a human-readable name for the domain. This name appears in the Fastly web interface.
6. Click the **Create** button to save the domain. The new domain appears in the list of domains.
7. Click the **Activate** button to deploy your configuration changes.

Once you've set up free TLS, you'll be able to access your host domain via the `https://<name>.global.ssl.fastly.net/` URL. You won't need to add CNAME records (/guides/basic-setup/adding-cname-records) to use the shared domain certificate.

Using HTTP/2 with free TLS

Your `<name>.global.ssl.fastly.net` domain name currently supports HTTP/1.1 and will be updated to support HTTP/2 later in 2017. To test HTTP/2 today, you can use `<name>.freetls.fastly.net`, which is automatically made available for all Fastly services. For example, if you originally claimed `example.global.ssl.fastly.net` during setup, Fastly automatically created `example.freetls.fastly.net` with support for HTTP/2 and HTTP/1.1.

---

§ Support for App Transport Security (/guides/securing-communications/support-for-app-transport-security)

Apple uses App Transport Security (https://developer.apple.com/library/content/documentation/General/Reference/InfoPlistKeyReferen SW35) (ATS) to improve the security of connections (/guides/securing-communications/) between web services and applications installed on devices using iOS 9 or later, as well as OS X 10.11 (El Capitan) and later. Fastly is fully compliant with all ATS requirements. You shouldn't run into any issues supporting iOS or OS X users while using our service.

Results from the ATS diagnostics tool

We used Apple's ATS diagnostics tool to ensure that Fastly is compliant with all ATS requirements. You can review the output from the diagnostics tool below.

```
$ /usr/bin/nscurl --ats-diagnostics https://www.fastly.com
Starting ATS Diagnostics

Configuring ATS Info.plist keys and displaying the result of HTTPS loads to https://www.fastly.
com.
A test will "PASS" if URLSession:task:didCompleteWithError: returns a nil error.

Use '--verbose' to view the ATS dictionaries used and to display the error received in URLSessi
on:task:didCompleteWithError:.
================================================================================

Default ATS Secure Connection
---
ATS Default Connection

Result : PASS
---

================================================================================
Allowing Arbitrary Loads

---
Allow All Loads

Result : PASS
---

================================================================================

Configuring TLS exceptions for www.fastly.com

---
TLSv1.2

Result : PASS
---

---
TLSv1.1

Result : PASS
---

---
TLSv1.0

Result : PASS
---

================================================================================
```

```
Configuring PFS exceptions for www.fastly.com

---
Disabling Perfect Forward Secrecy

Result : PASS
---

================================================================================

Configuring PFS exceptions and allowing insecure HTTP for www.fastly.com

---
Disabling Perfect Forward Secrecy and Allowing Insecure HTTP

Result : PASS
---

================================================================================

Configuring TLS exceptions with PFS disabled for www.fastly.com

---
TLSv1.2 with PFS disabled

Result : PASS
---

---
TLSv1.1 with PFS disabled

Result : PASS
---

---
TLSv1.0 with PFS disabled

Result : PASS
---

================================================================================

Configuring TLS exceptions with PFS disabled and insecure HTTP allowed for www.fastly.com

---
TLSv1.2 with PFS disabled and insecure HTTP allowed

Result : PASS
---

---
```

```
TLSv1.1 with PFS disabled and insecure HTTP allowed

Result : PASS
---

---
TLSv1.0 with PFS disabled and insecure HTTP allowed

Result : PASS
---
```

§ TLS termination (/guides/securing-communications/tls-termination)

Identifying TLS terminated requests

To maintain optimal caching performance, Fastly uses a TLS terminator (https://en.wikipedia.org/wiki/TLS_termination_proxy) separate from the caching engine. This means, however, that the caching engine doesn't know that it was originally a TLS request. As a result, we set the `Fastly-SSL` header when fetching the content from your servers.

Because we set this header, you can check for its presence on your backend by doing something like:

```
if (req.http.Fastly-SSL) {
  set resp.http.X-Is-SSL = "yes";
}
```

and that should tell you if the request was a TLS request or not.

When using WordPress

If you're using Fastly TLS services with WordPress, you'll want to add a check for the `HTTP_FASTLY_SSL` header so that WordPress can build URLs to your CSS or JS assets correctly. Do this by placing a check in your `wp-config.php` file to override the SSL flag that is checked later:

```
if (!empty( $_SERVER['HTTP_FASTLY_SSL'])) {
  $_SERVER['HTTPS'] = 'on';
}
```

As usual, this must be placed anywhere before the `require_once` line with `wp-settings.php`.

Finding the original IP when using TLS termination

Because Fastly uses a TLS terminator (https://en.wikipedia.org/wiki/TLS_termination_proxy), separate from the caching engine for performance, the engine overwrites the original IP briefly due to the re-request to your origin servers once decrypted and causes anything that references the original IP to show up as 127.0.0.0/8 IPs. To find the original IP via VCL:

- use `req.http.Fastly-Client-IP` if you're using shielding
- use `client.ip` if you're not using shielding or if you're building an ACL

Fastly also sends along the client IP to the origin in a HTTP header, `Fastly-Client-IP`, which can be used by server software to adjust as needed.

For more information about TLS-related issues, see our TLS guides (/guides/securing-communications/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

---

§ Fastly WAF logging (/guides/web-application-firewall/fastly-waf-logging)

Fastly provides a number of WAF-specific logging variables to help you monitor and identify potentially malicious traffic. These variables provide specific details about the actions Fastly WAF (/guides/web-application-firewall/web-application-firewall) performed on a request.

IMPORTANT: This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

OWASP rules

A single request can trigger multiple OWASP rules. By default, logging occurs in `vcl_deliver` or `vcl_log`. When logs are captured in `vcl_deliver` or `vcl_log`, it will show the last WAF rule triggered and the cumulative anomaly score.

waf_debug_log

The `waf_debug_log` subroutine allows logging of each OWASP rule triggered for a single request. To point your logging endpoint to this subroutine, update the logging placement parameter to `waf_debug` by running the following cURL command in a terminal application:

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version_id>/logging/<logging_integration>/<logging_name>' --data-binary '{"placement":"waf_debug"}'
```

- `waf_debug_log` accepts the logging format via the UI only
- `waf_debug_log` is called in `vcl_miss` and `vcl_pass`. The logging format can include request headers and WAF variables. Response headers will result in an error message.

We recommended creating a `request_id` header to track a single request through multiple OWASP rules:

```
set req.http.x-request-id = digest.hash_sha256(now randomstr(64) req.http.host req.url req.http.Fastly-Client-IP server.identity);
```

Using WAF-specific variables

Fastly provides a number of WAF-specific logging variables to help you monitor and identify potentially malicious traffic. These variables provide specific details about the actions Fastly WAF performed on a request:

- **Whether or not Fastly WAF inspected a request.** Fastly WAF only inspects traffic that is forwarded to your origin server (e.g., MISS or PASS requests for content that is not already cached).
- **Whether or not a rule matched the request.** When Fastly WAF inspects a request, it checks to see if the request matches any of the rules in your rule set.
- **The severity of the rule that matched.** If the request matches a rule, the log indicates the severity of the rule.
- **The action taken, if any.** If the request matches a rule or OWASP threshold, the log indicates whether Fastly WAF simply logged the request or blocked it.

You can use the following variables to examine Fastly WAF log events.

Variable	Description
`waf.executed`	A response header indicating if WAF was executed or not. Appears as `1` (true) when executed or `0` (false) when not.

Variable	Description
`waf.blocked`	Set to true when the request matches and the specific rule or OWASP threshold requirement is configured to block (/guides/web-application-firewall/managing-fastly-waf#blocking-requests). Will appear in log files as `1` (true) when blocked or `0` (false) when logged but not blocked.
`waf.logged`	In monitoring mode, set to `true` when the request matches and that specific rule is configured to log. Will show up in the logs as `1` (true) or `0` (false). In active (blocking) mode, set to `true` when `waf.blocked` is also `true`. Will show up in the logs as `1` (true) or `0` (false).
`waf.failures`	A request exits the WAF rule set due to a failure to evaluate. Will show up in the logs as `1` (true) or `0` (false).
`waf.logdata`	Why (specifically) this rule matched. Includes the portion of the request that triggered the match, so it may look different depending on the rule.
`waf.message`	A message describing the generic condition this rule matched. For example, `SLR: Arbitrary File Upload in Wordpress Gravity Forms plugin`.
`waf.rule_id`	The rule ID for this rule.
`waf.severity`	The severity of the rule. `0` is the highest severity and `7` is the lowest severity. `99` indicates that severity is not applicable (e.g., the request did not match any rules).
`waf.anomaly_score`	Cumulative score returned if request triggers OWASP rules. See OWASP category score variables.
`waf.passed`	Indicates if the request doesn't match any rules in the WAF rule set. Will show up in the logs as `1` (true) or `0` (false). `waf.passed` is readable in `vcl_deliver` and `vcl_log`. It is not readable in `waf_debug_log`. The value is determined after the request has gone through the WAF rule set.

OWASP category score variables

As a request goes through the OWASP rules, it can trigger different rule IDs from different attack categories. OWASP category score variables track which categories were triggered and the scoring that contributed to the cumulative score. They can be used to get a sense of minimum, average, and maximum values for a specific attack category and set thresholds individually. When in active (block) mode, if a request exceeds the category threshold, it will be blocked.

- `waf.sql_injection_score`

- `waf.rfi_score`
- `waf.lfi_score`
- `waf.rce_score`
- `waf.php_injection_score`
- `waf.session_fixation_score`
- `waf.http_violation_score`
- `waf.xss_score`

§ Fastly WAF rule set updates and maintenance (/guides/web-application-firewall/fastly-waf-rule-set-updates-maintenance)

Fastly provides rule set updates to the Fastly WAF (/guides/web-application-firewall/web-application-firewall) in a prompt manner to help protect customers against attacks.

For OWASP and Trustwave rules changes we use the following process:

1. We regularly review the rule changes as they happen in both the OWASP Core Rule Set and the Trustwave Rule Set.
2. We translate the rules into Varnish Configuration Language (VCL) (/guides/vcl/guide-to-vcl) to run inside our cache nodes.
3. We test the rules in our platform to ensure they perform adequately. We try to maximize performance and rule efficacy while reducing false positives.
4. We correct bugs, if any are found.
5. We propagate the rule set changes to our platform worldwide.
6. Finally, we will provide customers with a notification and instructions on how to make rule updates.

❗ **IMPORTANT:** This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

Rule set maintenance

The following table describes updates and changes to the provided rule sets:

ID	Date	Type of Change	A s
`2YXlqZJQxMkWyAjM4kggR3`	2017/11/13	• Global update to OWASP 3.0.2 CRS release • Update Trustwave rules to latest available • Introduce new Fastly internal rule 10040 to block any HTTP POST body greater than 2 kibibytes in size.	
`2vyJNHO7fngQYJXU8UGUY6`	2017/10/11	• Updates to rule 932140 to account for SAML false positives in Windows • Reintroduction of missing transforms on some OWASP rules • Introduction of Fastly internal rule to protect against CVE-2017-9805 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-9805)	
`4Z09wgjp7do8NrOIzlckFS`	2017/08/15	• Reintroduction of individual threshold variables: `http_violation_score_threshold`, `lfi_score_threshold`, `php_injection_score_threshold`, `rce_score_threshold`, `rfi_score_threshold`, `session_fixation_score_threshold`, `sql_injection_score_threshold`, `xss_score_threshold` • Removal of unused threshold variables: `brute_force_counter_threshold`, `dos_counter_threshold`, `outbound_anomaly_score_threshold`, `trojan_score_threshold` • Additional bug fixes in OWASP rule set	

ID	Date	Type of Change	A s
`39EE4tZnEM9Q8hxFJMHYU5`	2017/04/27	• Global update to the OWASP CRS 3.0 rule set • New Fastly rule for the February 2017 Wordpress Code Injection (https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html) • New Fastly rule for the March 2017 Apache Struts RCE exploit (http://blog.talosintelligence.com/2017/03/apache-0-day-exploited.html) • Updated Trustwave content inspection rules	

Updating to the newest rule set

Follow these instructions to update a WAF to use the newest rule set.

Reviewing the current rule set

Before updating your WAF to a new rule set, we recommend that you record the value of your WAF's currently active rule set. You can use this information to revert your WAF to its previous state.

Run the following cURL command in a terminal application to find the currently active rule set:

```
curl -s -H Fastly-Key: <your Fastly API token> -H Accept:application/vnd.api+json \
     https://api.fastly.com/service/<your Fastly service ID>/version/<your service version number>/wafs/<your WAF ID>
```

★ **TIP:** You can use this API endpoint (/api/waf#waf_firewall_989adbdf8ec7257d37283e21ae2391f2) to find your WAF's ID.

The output from the cURL command is shown below. In the `relationships` object, notice that this WAF is using `<ID of your active configuration set>`. Remember the ID.

```
{
    "data": {
        "attributes": {
            "last_push": null,
            "prefetch_condition": null,
            "response": null,
            "version": "1"
        },
        "id": "<your WAF ID>",
        "relationships": {
            "configuration_set": {
                "data": {
                    "id": "<ID of your active configuration set>",
                    "type": "configuration_set"
                }
            }
        },
        "type": "waf"
    }
}
```

Changing the rule set version

Follow these instructions to change the rule set version for a WAF:

1. Find the ID of the new rule set version you want to use in the rule set maintenance section.
2. On your computer, create a new file called `updated_relationship.json`.
3. Copy and paste the following JSON into the file, replacing `<your rules ID>` with the ID of the rule set version you want to use:

   ```
   {
       "data": {
           "id": "<your WAF ID>",
           "relationships": {
               "configuration_set": {
                   "data": {
                       "id": "<your rules ID>",
                       "type": "configuration_set"
                   }
               }
           },
           "type": "waf"
       }
   }
   ```

4. Save the changes to the `updated_relationship.json` file.

5. In the directory you saved the file, run the following cURL command in a terminal application to change the rule set version for a WAF:

```
curl -s -X PATCH -H Fastly-Key: <your Fastly API token> -H Accept:application/vnd.api+json
\
  -H Content-Type:application/vnd.api+json -d @updated_relationship.json \
  https://api.fastly.com/service/<your Fastly service ID>/version/<your service version nu
mber>/wafs/<your WAF ID>
```

6. Changing the rule set version for a WAF can take some time. Run the following cURL command in a terminal application to monitor the status of the process:

```
curl -s -H Fastly-Key: <your Fastly API token> -H Accept:application/vnd.api+json \
      https://api.fastly.com/service/<your Fastly service ID>/version/<your service versio
n number>/wafs/<your WAF ID>
```

   The process is complete when the output displays the ID of the new rule set version.

Updating to the latest rules

After you've verified that the rule set for the WAF has successfully been changed, follow these rules to update your WAF with the latest rules:

1. Run the following cURL command in a terminal application to update the rule set:

```
curl -s -X PATCH -H Fastly-Key: <your Fastly API token> -H Accept:application/vnd.api+json
\
  -H Content-Type:application/vnd.api+json -d '{"data":{"id":"<your WAF ID>","type":"rules
et"}}' \
  https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/ruleset
```

   The response will look like this:

```
{
    "data": {
        "id": "WAF_ID",
        "type": "ruleset"
    },
    "links": {
        "related": {
            "href": "https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WA
F ID>/update_statuses/<update status ID>"
        }
    }
}
```

2. Updating the WAF with the latest rules can take some time. Using the URL in the response in the previous step, run the following cURL command in a terminal application to monitor the status of the process:

```
curl -s -H Fastly-Key: FASTLY_API_TOKEN -H Accept:application/vnd.api+json \
https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/update_statuses/<update status ID>
```

The response for the `waf_update_status` will have a `status` of `complete` when the process is complete.

```
{
    "data": {
        "attributes": {
            "completed_at": "2017-04-05 18:47:28 UTC",
            "created_at": "2017-04-05 18:47:27 UTC",
            "message": null,
            "status": "complete",
            "updated_at": "2017-04-05 18:47:28 UTC"
        },
        "id": "<update status ID>",
        "type": "waf_update_status"
    }
}
```

Reverting to a previous rule set version

If a WAF rule set update doesn't go as planned, you can revert to the previous rule set version. Using the previous rule set ID you recorded in the reviewing the current rule set section, follow the instructions in changing the rule set version and updating to the latest rules.

---

§ Managing the Fastly WAF (/guides/web-application-firewall/managing-fastly-waf)

The Fastly WAF (/guides/web-application-firewall/web-application-firewall) provides rules that detect and block potential attacks. The rules are collected into a policy and deployed within your Fastly service at the edge.

IMPORTANT: This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

Inspecting the Fastly WAF rule set

You can inspect your Fastly WAF rule set at any time. By making an API call (/api/waf#waf_ruleset_5e37b65346b88d394dcf3ba6b07e35d4), you can download all of the data associated with your Fastly WAF rules. To inspect your Fastly WAF rule set, run the following cURL command in a terminal application:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/ruleset | perl -pe 's/\\n/\n/g'
```

NOTE: The `| perl -pe 's/\\n/\n/g'` is optional and can assist with formatting.

Inspecting the VCL of a WAF rule

To inspect the VCL of a specific Fastly WAF rule, run the following cURL command in a terminal application:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/wafs/<your WAF ID> /rules/<rule_id>/vcl
```

See the API documentation (/api/waf#waf_rule_bc8528ca71eac3834b010c8a288b0588) for more information.

Blocking requests

When you start using Fastly WAF for the first time, all rules are set to `log` status to minimize false positives. We recommend you monitor the logs (/guides/web-application-firewall/fastly-waf-logging) for a minimum of two weeks to make sure that the rules will not block legitimate requests to your web application. Requests will not be blocked until you switch one or more rules from `log` to `block` status.

Changing the status of rules

To change a rule from `log` status to `disabled` or `block` status, inspect your rule set or review (/guides/web-application-firewall/fastly-waf-logging) your logs to find the `waf.rule_id` variable (/guides/web-application-firewall/fastly-waf-logging#using-waf-specific-variables). Then, run the following cURL command in a terminal application for each rule:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN' -X PATCH -d '{"data": {"id": "<your WAF ID>-<WAF rule ID>", "type": "rule_status", "attributes":{ "status": "block"}}}' -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/rules/<WAF rule ID>/rule_status
```

To change the status of a group of rules, use a filter-tag (e.g., `application-WordPress`, `language-html`, or `OWASP`) by running the following cURL command in a terminal application:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN' -X POST -d '{"data": {"id": "<your WAF ID>", "type": "rule_status", "attributes": {"name": <tag>, "status": "block"}}}' -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/rule_statuses
```

ℹ **NOTE:** When changing rule statuses for a group of rules using a filter-tag, the above API call will preserve the status of any disabled rules updated individually. If all rules under the filter-tag should be forced to have a `log` or `block` state, add the parameter `force:true` under attributes in the request body.

See the API documentation (/api/waf#waf_rule_status_e71d08db43cf2e8dce2e73194b071021) for more information. When you've finished setting rules to `block` status, you'll need to activate the changes.

ℹ **NOTE:** If you need to enable more than 1,000 rules, contact our customer support team at support@fastly.com (mailto:support@fastly.com).

OWASP Configuration

OWASP blocking is dependent on the following:

- All OWASP rules (excluding rules changed from `log` to `disabled` mode) set to `block` mode.
- Threshold limits set for the cumulative score and attack categories.

If a request triggers OWASP rules, it returns attack category scores and a cumulative score. If any of the final scores exceed the threshold limit and the OWASP rules are in block mode, Fastly sends the custom error response to the user.

Viewing OWASP settings

To view your OWASP settings, run following cURL command in a terminal application:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<service_id>/wafs/<your WAF ID>/owasp
```

The cumulative anomaly score is displayed in the `inbound_anomaly_score_threshold` field.

Changing OWASP settings

To change any OWASP settings object, run the following OWASP update command (/api/waf#waf_owasp_e3d26089888473ea1f7d48016a518ee9) in a terminal application:

```
curl -X PATCH -v -H "Content-Type: application/vnd.api+json" -H "Accept: application/vnd.api+js
on" -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/wafs/<waf_id
>/owasp -d '{"data": {"attributes":{"inbound_anomaly_score_threshold":"50"}, "id":"<owasp_id>",
"type":"owasp"}}'
```

When you've finished modifying OWASP settings, you'll need to activate the changes.

Activating changes

After you modify the status of one or more rules, you must activate the changes by running the following cURL command in a terminal application:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN' -X PATCH -d '{"data": {"id": "<your WAF ID>", "type": "r
uleset"}}' -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/ID/wafs/I
D/ruleset
```

See the API documentation (/api/waf#waf_ruleset_3c076195f3616cf75dc9cf274916a768) for more information.

Rules are versionless. Any changes to the rules will become effective after you run the command shown above. You won't need to activate a new version of your service (/guides/basic-setup/working-with-services#editing-and-activating-versions-of-services) to have the changes take effect.

---

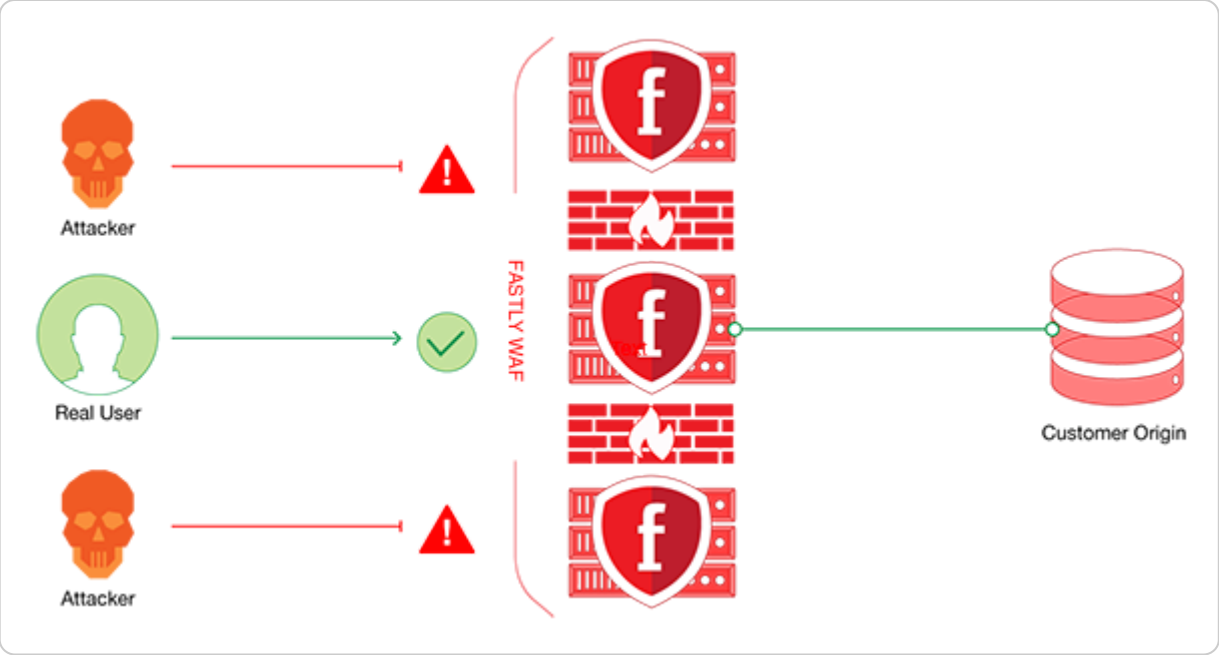
§ Web Application Firewall (WAF) (/guides/web-application-firewall/web-application-firewall)

Fastly offers a web application firewall (WAF) security service that allows you to detect malicious request traffic and log or log and block (/guides/web-application-firewall/fastly-waf-logging) that traffic before it reaches your web application. The Fastly WAF provides rules that detect and block potential attacks (/guides/web-application-firewall/managing-fastly-waf#blocking-requests). The rules are collected into a policy and deployed within your Fastly service at the edge. To get started, email our sales team (mailto:sales@fastly.com) for product information.

IMPORTANT: This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

How the Fastly WAF works

The Fastly WAF is designed to protect production web applications running over HTTP or HTTPS against known vulnerabilities and common attacks such as cross-site scripting (XSS) and SQL injection. The Fastly WAF can provide a layer of protection logically positioned at the client edge of your distributed application to detect and block malicious activity from exploiting vulnerabilities in web applications and APIs.



Like traditional network firewall appliances, Fastly WAF uses predetermined security rules to monitor and control incoming traffic to your web application. A network firewall works at the IP level and often blocks IP addresses from untrusted networks, preventing them from gaining access to a private network. Unlike firewalls at the network or transport layer level, the Fastly WAF works by analyzing web traffic primarily at the HTTP application layer. It reads all HTTP(S) headers and the post body of the HTTP(S) requests that it inspects and runs them through a rule set selected for your service environment.

Fastly provides a default WAF rule set to which you can add additional rule sets to help protect against application-specific attacks. Once the Fastly WAF is enabled for a version of your service (/guides/basic-setup/working-with-services.html#editing-and-activating-versions-of-services), you can change the status of any individual rule to logging, blocking, or disabled mode (/guides/web-application-firewall/managing-fastly-waf#blocking-requests). Rule changes are versionless and become effective immediately.

NOTE: The Fastly WAF only works when traffic is directed through it. Make sure that you've signed up (/guides/basic-setup/sign-up-and-create-your-first-service#sign-up-at-fastlycom) for Fastly, created a service (/guides/basic-setup/sign-up-and-create-your-first-

service#create-your-first-service), and added a CNAME DNS record (/guides/basic-setup/adding-cname-records) for your domain name to direct traffic to Fastly and through the Fastly WAF.

Enabling the Fastly WAF

Enabling Fastly WAF doesn't require modifications to your web application or origin servers. Contact our sales team (mailto:sales@fastly.com) to get started.

Refining the default WAF policy once it's enabled

Once you purchase the Fastly WAF, our customer support team (mailto:support@fastly.com) will enable it with the default WAF policy for any service you've provided a service ID for. They will then work closely with you on additional configuration refinements, including:

- setting up a logging endpoint,
- selecting rule sets and a prefetch condition, and
- optionally customizing the request responses.

You can then begin monitoring logs (/guides/web-application-firewall/fastly-waf-logging) to determine which requests to your origin are legitimate and which you should consider blocking (/guides/web-application-firewall/managing-fastly-waf#blocking-requests) to protect your origin.

Setting up a logging endpoint

To begin monitoring requests for potential malicious activity, set up remote logging (/guides/streaming-logs/setting-up-remote-log-streaming) so you can log WAF variables (/guides/web-application-firewall/fastly-waf-logging#using-waf-specific-variables). You can use an existing logging endpoint or add a new endpoint specially for Fastly WAF. You'll use the information provided in the logs to monitor WAF events (/guides/web-application-firewall/fastly-waf-logging).

Selecting rule sets

Fastly provides a default WAF rule set that is based on ModSecurity Rules from Trustwave SpiderLabs (https://www.trustwave.com/Products/Application-Security/ModSecurity-Rules-and-Support/) and the OWASP Top Ten (https://www.owasp.org/index.php/Top_Ten). The default rule set is designed to help you monitor web application traffic for a wide range of common attacks.

Fastly adds a default prefetch condition (`!req.backend.is_shield`) for the WAF policy. This ensures that the Fastly WAF inspects traffic to the origin and accounts for whether or not a service has shielding (/guides/performance-tuning/shielding) configured.

You can modify the prefetch condition. For example, you could update the prefetch statement to run the WAF rule set on origin traffic and requests from IP addresses that aren't whitelisted:

```
curl -v -X PUT https://api.fastly.com/service/<your Fastly service ID>/version/<version_id>/con
dition/Waf_Prefetch -H "Fastly-Key: FASTLY_API_TOKEN" -H "Content-Type: application/json" -d
'{"statement":"!req.backend.is_shield && !(client.ip ~ whitelist)"}' -H "Accept: application/js
on"
```

Fastly can add additional rule sets for specific applications or technologies (e.g., WordPress, Drupal, PHP, .Net). Keep in mind that adding additional rule sets can increase latency for requests being evaluated against the published WAF policy.

Once you've selected rule sets, Fastly will maintain rules (/guides/web-application-firewall/fastly-waf-rule-set-updates-maintenance) sourced by Fastly to keep them current. However, you'll need to notify us if you modify the applications or technologies that are present at the origin.

Customizing the response

Fastly's customer support team creates a custom response and assigns an HTTP status code for all requests that Fastly WAF blocks. If you've configured Fastly WAF to block requests (/guides/web-application-firewall/managing-fastly-waf#blocking-requests), that response will be served directly from the cache when a request matches a rule. If you would like to customize the response, use the web interface to change (/guides/basic-configuration/responses-tutorial#creating-a-response) the following:

- **MIME Type:** The content type of the response.
- **Response:** The content served when delivering the response.

⚠ **WARNING:** Do not modify the **Status** or **Description** of the Fastly WAF response that customer support creates for you.

Disabling Fastly WAF for your service

Contact our customer support team at support@fastly.com (mailto:support@fastly.com) to disable the Fastly WAF for your service.

Limitations

All WAF products that exist today have several limitations:

- **False positives:** Any WAF can mistake good traffic for bad. This is why we strongly recommend that you monitor your logs (/guides/web-application-firewall/fastly-waf-logging) for a minimum of two weeks before blocking traffic (/guides/web-application-

firewall/managing-fastly-waf#blocking-requests). You don't want start blocking traffic with rules that are generating false positives.

- **DNS configuration:** A WAF only works when traffic is directed through it. It cannot protect against malicious requests that are sent to domain names or IP addresses that are not specified in your WAF configuration.
- **Effective rule sets:** A WAF is only as effective as its rule sets. You should add rule sets as necessary to protect your specific web application.
- **Custom application vulnerabilities:** If attackers discover a vulnerability unique to your application or the technologies you use, and your WAF configuration does not have a rule to protect against exploits for that particular vulnerability, it will not be able to protect your application in that instance. Customer support can work with you to add additional rule sets to help protect against these types of attacks. If you need more protection than the rule sets provide, customer support can work with you to create custom VCL to help block malicious requests.
- **Inspection of HTTP and HTTPS traffic only:** A WAF only inspects HTTP or HTTPS requests. It will not process any TCP, UDP, or ICMP requests.

LA limitations

The Fastly WAF is part of a limited availability release (/guides/fastly-product-lifecycle/#limited-availability) and it has the following limitations:

- Inspecting the WAF rule set (/guides/web-application-firewall/managing-fastly-waf#inspecting-the-fastly-waf-rule-set) is challenging due to formatting issues with cURL.
- Changes are managed via the API (/api/waf).

Security products note

❗ **IMPORTANT:** To ensure your web application only receives traffic from your WAF-enabled Fastly service, we strongly recommend you configure TLS client authentication (/guides/basic-configuration/connecting-to-origins#specifying-a-tls-client-certificate-and-key) for that service and whitelist Fastly's assigned IP ranges (/guides/securing-communications/accessing-fastlys-ip-ranges).

No security product, such as a WAF or DDoS mitigation product, including those security services offered by Fastly, will detect or prevent all possible attacks or threats. Subscribers should maintain appropriate security controls on all web applications and origins, and the use of Fastly's security products do not relieve subscribers of this obligation. Subscribers should test and validate the effectiveness of Fastly's security services to the extent possible prior to deploying these services

in production, and continuously monitor their performance and adjust these services as appropriate to address changes in the Subscriber's web applications, origin services, and configurations of the other aspects of the Subscriber's Fastly services.

---

§ IP geolocation variables: Migrating to the new dataset (/guides/migrations/migrating-geolocation-variables-to-the-new-dataset)

Fastly's IP geolocation variables (/guides/vcl/new-geolocation-related-vcl-features) are now based on a new IP geolocation dataset. Following Fastly's feature retirement policy (/guides/fastly-product-lifecycle/index#product-or-feature-retirement), we'll continue to support variables that use the older version of the geolocation dataset until all of our customers have had time to migrate their service configurations to the newer version. As you migrate your configurations, keep the following important considerations in mind.

Namespaces differ between versions

The old version of the IP geolocation variables exist in the `geoip` namespace. The new version of these variables exist in the `client.geo` namespace and the Autonomous System (AS) variables exist in the `client.as` namespace.

Results for IPv6 addresses will only be returned for `client.geo` and `client.as` namespaces.

Geolocation data may be different

The data returned for a given IP address may be different between the dataset versions, especially at the city level. While it's possible to migrate configurations by replacing the older `geoip.*` namespace with `client.geo.*`, we recommend you carefully review any business logic that may rely on this data, especially if it's implemented in VCL or if the values are exposed via HTTP headers or real-time streaming logs.

In particular, understand that:

- The IP geolocation datasets are sourced from different vendors, each with different conventions for textual values. For example, `client.geo.city` and `client.geo.country_name` in the new dataset exist as lowercase ASCII values whereas the

values returned for the same fields in older dataset are mixed case.

- The `client.geo.region` field contains ISO 3166-2 (https://en.wikipedia.org/wiki/ISO_3166-2) region codes but the `geoip.region` field contains FIPS 10-4 (https://en.wikipedia.org/wiki/FIPS_10-4) region codes. The FIPS 10-4 standard was withdrawn in 2014.

---

§ Acquia Cloud (/guides/integrations/acquia-cloud)

To use Acquia Cloud as an origin, you must sign up for both an Acquia Cloud subscription and Fastly services and connect the two.

Sign up for an Acquia Cloud subscription

1. Using a web browser, navigate to the Acquia Cloud signup page (https://www.acquia.com/free).
2. Select the Acquia Cloud Free option. The account subscription form appears.
3. Fill out the form and click **Create** to sign up for a subscription and start Acquia's automated site creation process.

The automated portion of the Acquia subscription process can take three to five minutes to complete. You'll know the entire process ends successfully when you see the checkmark appear next to the word "Done."

Check for domain alias conflicts

Ensure that your new site domain does not have a conflicting alias by running the `host` command on a command line. For example, the `host` command for the Test-Example-2 domain would be:

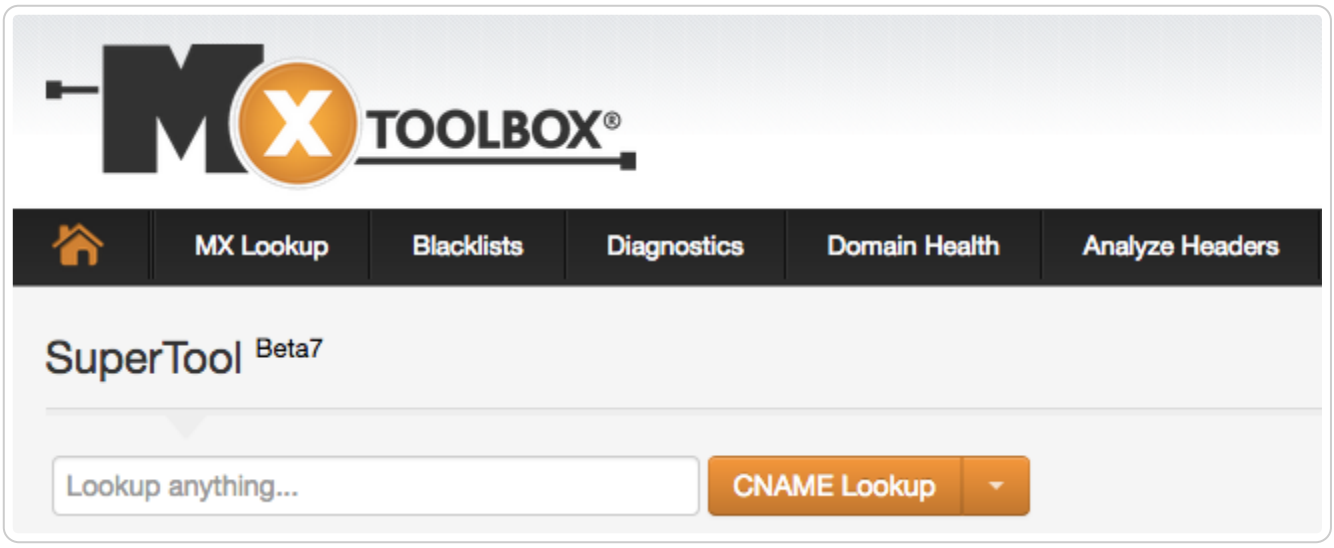
`host test-example-2.devcloud.acquia-sites.com`.

and would produce the following sample output:

`test-example-2.devcloud.acquia-sites.com has address 127.0.0.1`

Determine your CNAME

1. Using a web browser, navigate to the MX Toolbox SuperTool (http://mxtoolbox.com/SuperTool.aspx).



2. In the **Lookup anything** field, type the name of your website domain.
3. From the menu to the right of the field, select **CNAME Lookup**. The domain name and canonical name appear below the field.
4. Save this information to use when you sign up for Fastly services.

Sign up for Fastly CDN services

1. Using a web browser, navigate to the Fastly signup page (https://www.fastly.com/signup) and sign up for a Fastly account. The system sends a confirmation email to the address you specified during signup.
2. Verify your new Fastly account by clicking on the verification link sent to the email address used to sign up for Fastly service.
3. Log in to the Fastly web interface and complete your account configuration (/guides/basic-setup/sign-up-and-create-your-first-service).

Complete the integration

Once you have completed the signup and configuration steps, send an email to acquia@fastly.com (mailto:acquia@fastly.com) to complete the integration. Fastly will need to know:

- the email address associated with your new Fastly account and
- whether or not you require TLS (/guides/detailed-product-descriptions/tls-service-options) for your customer-facing domain.

Keep in mind that Fastly and Acquia both run a Varnish Cache (https://varnish-cache.org). In order to properly configure your service, Fastly needs to make a few modifications during the final setup process to ensure compatibility between the two. In this final setup process, Fastly runs a script

(https://gist.github.com/vvuksan/66cc45e09812fbf90808) to automatically provision and configure your Fastly account. This script will:

- create an Acquia-specific service within your Fastly account,
- add your Acquia origin to the new service,
- add your end-user-facing domain to the new service, and
- add caching configurations to your service to optimize content delivery.

As soon as the configuration on Fastly's side is complete, you will receive email notification that you can change your CNAME records (/guides/basic-setup/adding-cname-records) to point at the appropriate corresponding Fastly endpoints. As soon as the CNAME process is complete, you'll be ready to start using Acquia Cloud as an origin for Fastly services.

> This article describes an integration with a service provided by a third party. Please see our note on integrations (/guides/integrations/).

---

§ Amazon S3 (/guides/integrations/amazon-s3)

Amazon S3 (http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html) public and private buckets can be used as origins with Fastly.

Using Amazon S3 as an origin

To make your S3 data buckets available through Fastly, follow the steps below.

Creating a new service

Follow the instructions for creating a new service (/guides/basic-setup/working-with-services#creating-a-new-service). You'll add specific details about your origin when you fill out the **Create a new service** fields:

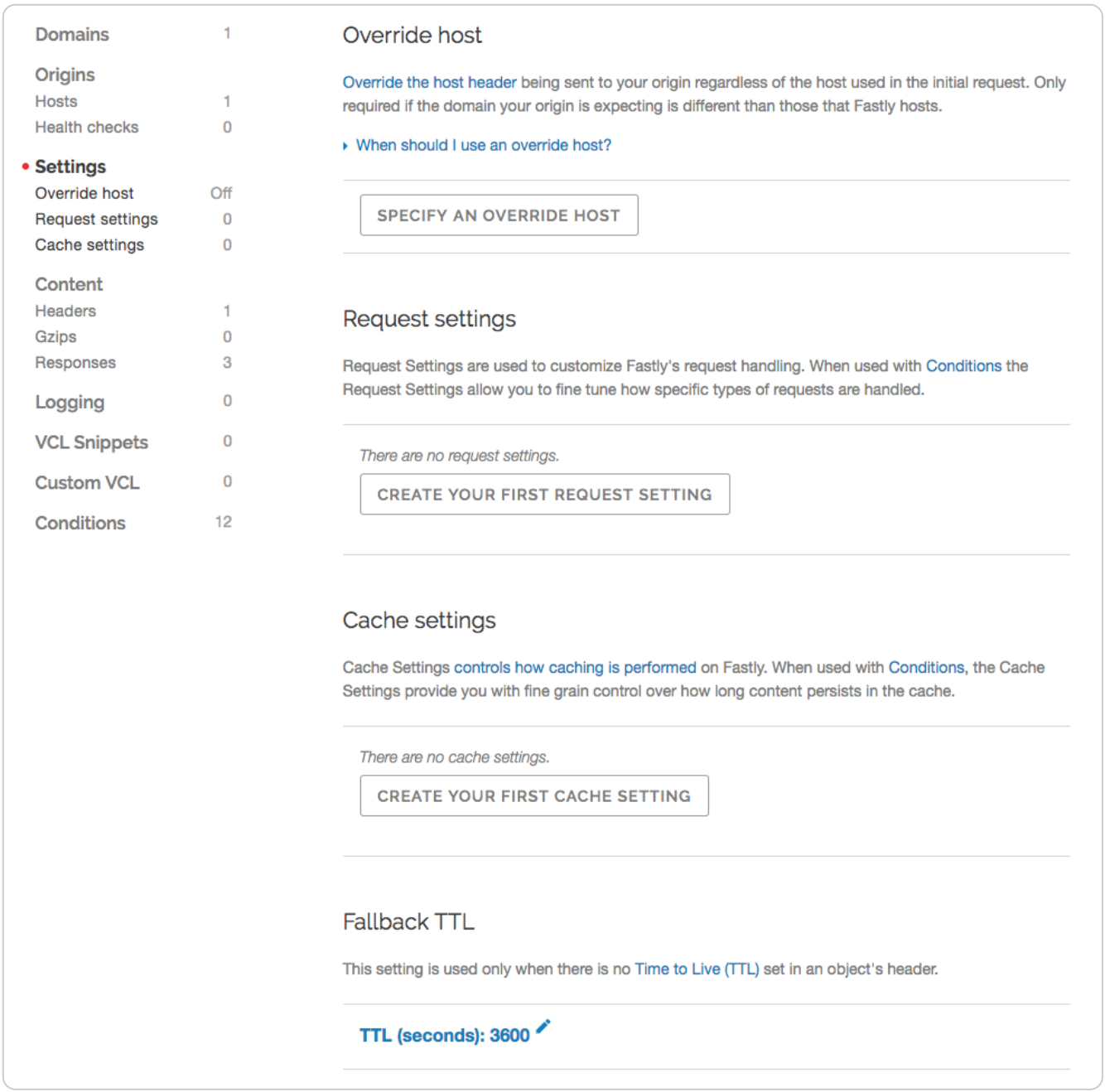
- In the **Name** field, type any descriptive name for your service.
- In the **Domain** field, type the hostname you want to use as the URL (e.g., `cdn.example.com`).
- In the **Address** field, type `s3.amazonaws.com`. If you are using a non-standard S3 region (anything other than us-east), you must include that region in the server address field (e.g., `s3.us-west-2.amazonaws.com`).

- In the **Transport Layer Security (TLS)** area, leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
- In the **Transport Layer Security (TLS)** area, type `s3.amazonaws.com` in the **Certificate hostname** field.

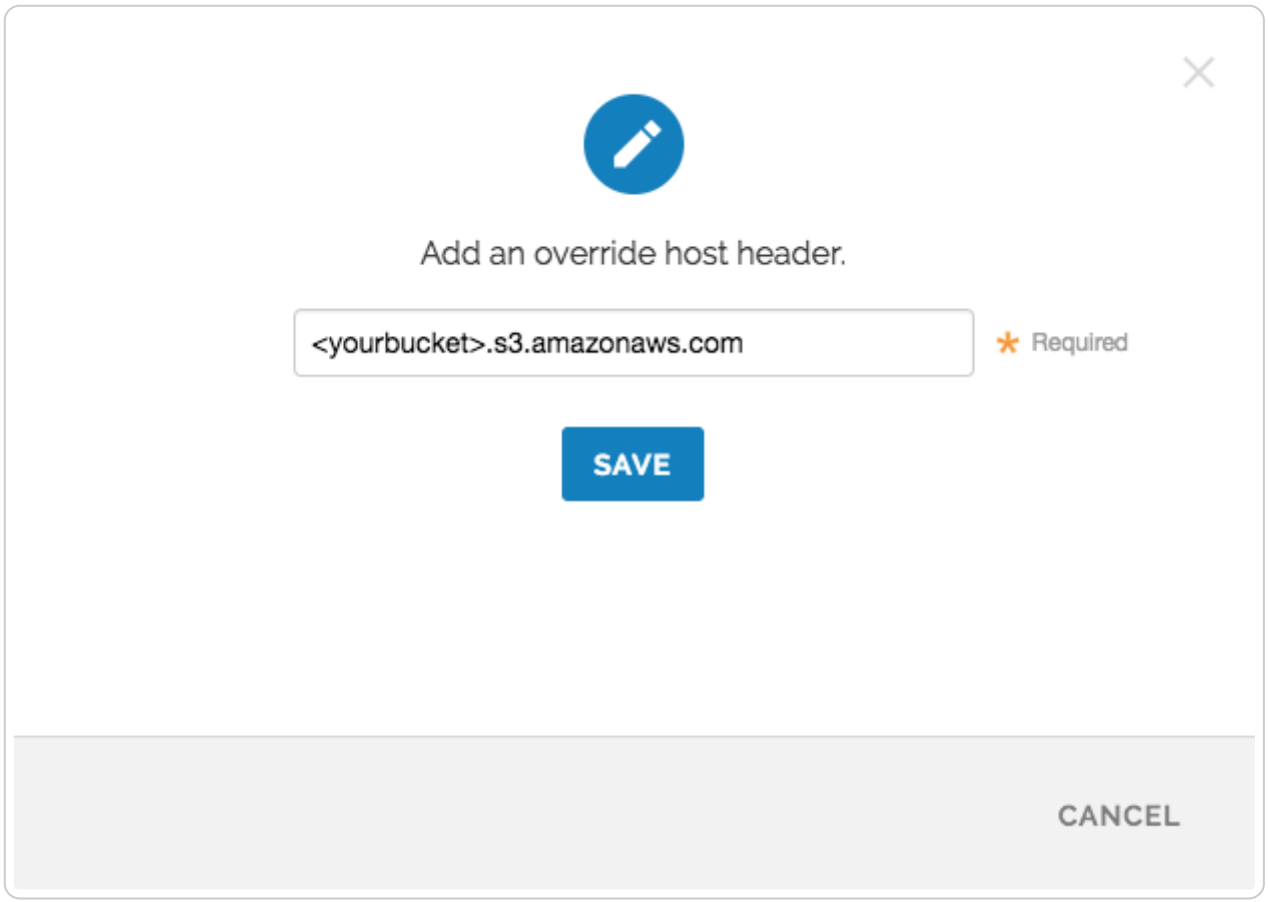
Setting the default host

Once the new service is created, set the default host to `<yourbucket>.s3.amazonaws.com` by following the steps below:

1. From the service menu, select the appropriate service.
2. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
3. Click the **Settings** link. The Settings page appears.



4. Click the **Specify an Override Host** button. The Add an override host header window appears.



5. Type the hostname of your S3 bucket. For example, `<yourbucket>.s3.amazonaws.com`.

> ★ **TIP:** If you're using S3 to host a static website, the hostname is `<bucket_name>.s3-website-<aws_region>.amazonaws.com`, where `<bucket_name>` is the name of your S3 bucket, and `<aws_region>` is the name of the AWS region (e.g., `mygreatbucket.s3-website-us-east-1.amazonaws.com`).

6. Click the **Save** button.
7. Click the **Activate** button to deploy your configuration changes.

Testing your results

By default, we create DNS mapping called **yourdomain.global.prod.fastly.net**. In the example above, it would be `cdn.example.com.global.prod.fastly.net`. Create a DNS alias for the domain name you specified (e.g., CNAME `cdn.example.com` to `global-nossl.fastly.net`).

Fastly will cache any content without an explicit `Cache-Control` header for 1 hour. You can verify whether you are sending any cache headers using cURL. For example:

```
$ curl -I opscode-full-stack.s3.amazonaws.com

HTTP/1.1 200 OK
x-amz-id-2: ZpzRp7IWc6MJ8NtDEFGH12QBdk2CM1+RzVOngQbhMp2f2ZyalkFsZd4qPaLMkSlh
x-amz-request-id: ABV5032583242618
Date: Fri, 18 Mar 2012 17:15:38 GMT
Content-Type: application/xml
Transfer-Encoding: chunked
Server: AmazonS3
```

In this example, no cache control headers are set so the default TTL will be applied.

Enhanced cache control

If you need more control over how different types of assets are cached (e.g., Javascript files, images), check out our Amazon S3 configuration (/guides/tutorials/cache-control-tutorial#amazon-s3-configuration) in our Cache Control tutorial.

Using an Amazon S3 private bucket

To use an Amazon S3 private bucket with Fastly, follow the instructions below.

Before you begin

Be sure you've already made your S3 data buckets available to Fastly by pointing to the right S3 bucket and setting your origin to port 443. This needs to be done before authenticating.

Be sure you've got the AWS access key ID, AWS secret key ID, and AWS Bucket name on hand. The Amazon S3 Authorization header takes the following form:

```
Authorization: AWS `_AWSAccessKeyId_`:`_Signature_`
```

You'll need the following information from your developer Amazon account:

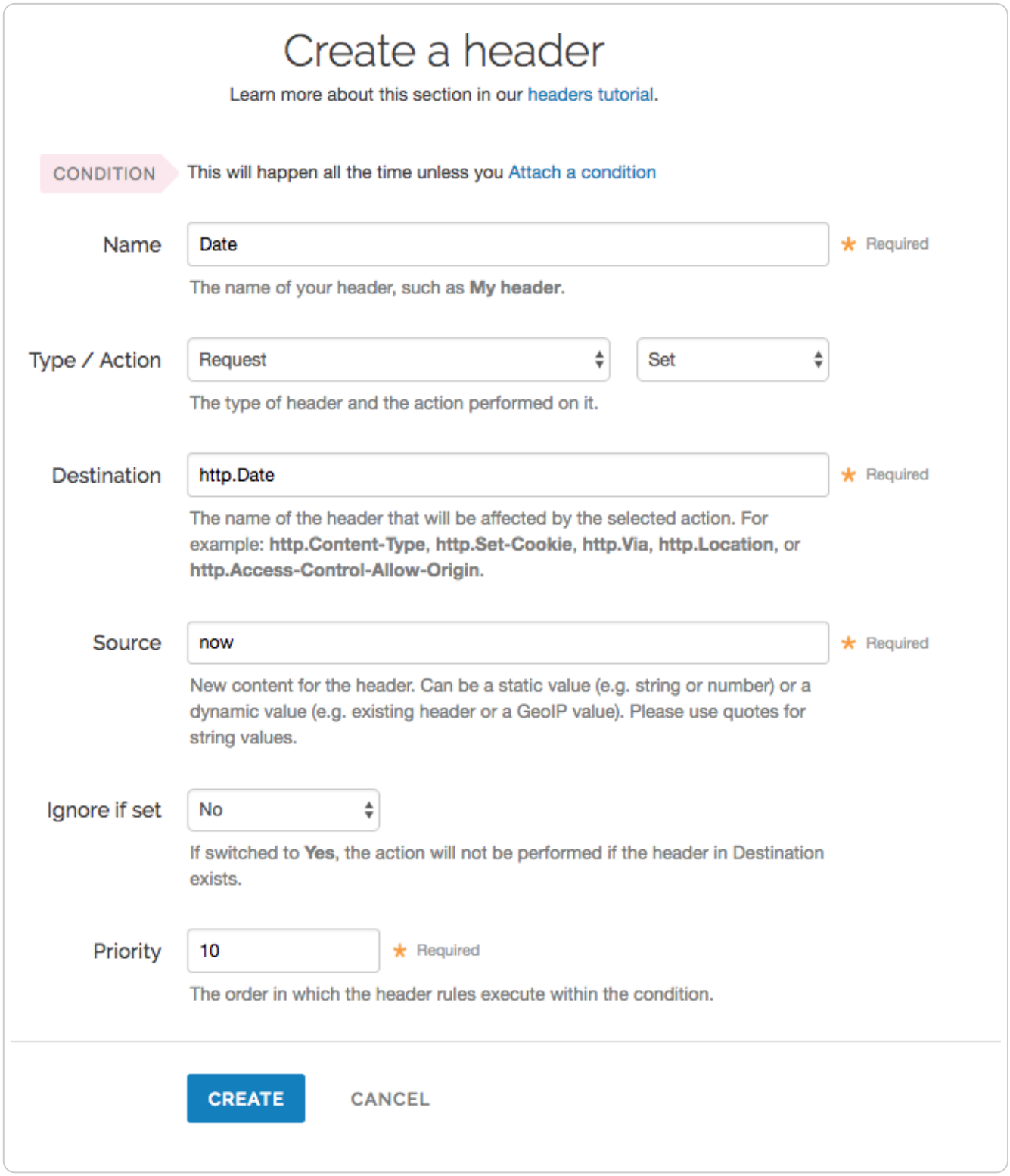
1. The **AWS access key ID** and **AWS secret access key**. The AWS secret access key is issued when you register. If you don't have or remember your AWS secret access key, create a new AWS access key ID. The AWS secret access key will be displayed before disappearing again.
2. Your **AWS Bucket** name.

Setting up Fastly to use an Amazon S3 private bucket

In order to use an Amazon S3 private bucket with Fastly, create two headers (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses), a Date header (for use with the authorization Signature) and an Authorization header.

Create a Date header

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.

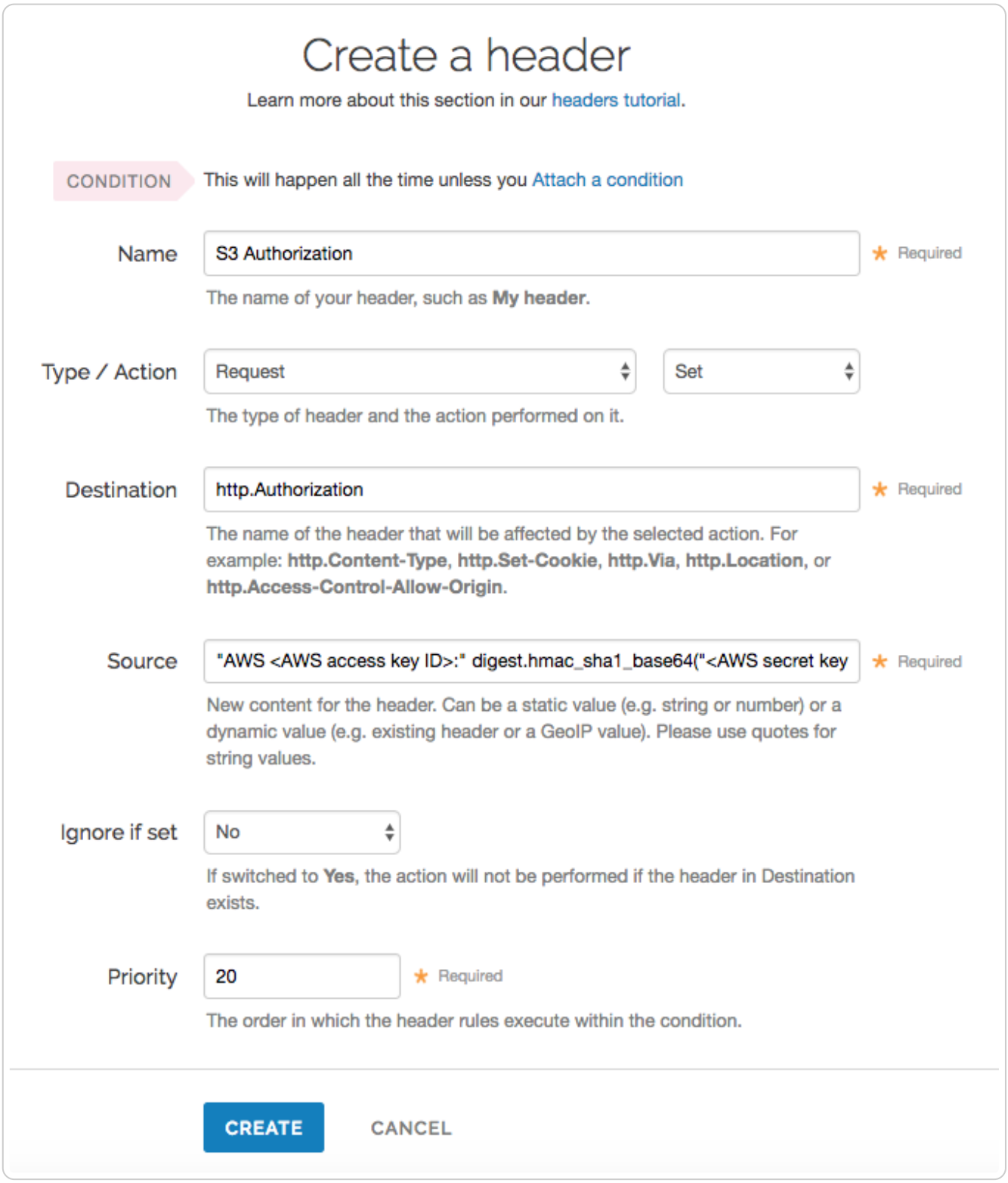


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Date`.
   - In the **Source** field, type `now`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `10`.
7. Click the **Create** button. A new Date header appears on the Content page. You will use this later within the Signature of the Authorization header.

Create an Authorization header

Next create the Authorization header with the specifications listed below.

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `S3 Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.

- In the **Priority** field, type `20`.

3. In the **Source** field, type the header authorization information using the following format:

```
"AWS <AWS access key ID>:" digest.hmac_sha1_base64("<AWS secret key ID>", if(req.request ==
= "HEAD", "GET", req.request) LF LF LF req.http.Date LF "/<AWS Bucket name>" req.url.path)
```

   replacing `<AWS access key ID>`, `<AWS secret key ID>`, and `<AWS Bucket name>` with the information you gathered before you began. For example:

```
"AWS JKCAUEFV2ONFFOFMSSLA:" digest.hmac_sha1_base64("P2WPSu68Bfl89j72vT+bXYZB7SjlOwhT4whqt
27", if(req.request == "HEAD", "GET", req.request) LF LF LF req.http.Date LF "/test123" re
q.url.path)
```

4. Click the **Create** button. The new Authorization header appears on the Content page.

A detailed look at the Source field

So what's going on in the Source field of the Authorization header? Here's the basic format:

`AWS<Access Key><Signature Function><key><message>`

It tells us the following:

Element	Description
`AWS`	A constant placed before the access key. It's always AWS.
`access key`	The access key ID from your Amazon developer's account. We used `JKCAUEFV2ONFFOFMSSLA` in this example.
`signature function`	The algorithm used to validate the key and message of the signature. We used `digest.hmac_sha1_base64(<key>, <message>)` in this example.
`key`	The secret key ID from your Amazon developer's account. We used `P2WPSu68BfI89j72vT+bXYZB7SjIOwhT4whqt27` in this example.
`message`	The UTF-8 encoding of the StringToSign. See the table below for a break down of each portion of the message.

The message that's part of the Source field in the Authorization header takes on this basic format:

`<HTTP-verb></n><Content-MD5>/n<Content-Type></n><Date></n><CanonicalizedAmzHeader></n><CanonicalizedResource>`

It tells us the following:

Element	Description

Element	Description
`HTTP-verb`	The REST verb. We use `req.request` in this example. We rewrite HEAD to GET because Varnish does this internally before sending requests to origin.
`/n`	A newline indicator constant. It's always /n.
`Content-MD5`	The content-md5 header value, used as a message integrity check. It's often left blank. We use `LF` (line feed) in this example.
`Content-Type`	The content-type header value, used to specify the MIME-type. It's often left blank. We use `LF` in this example.
`Date`	The date and time stamp. We use `req.http.Date` (which we created first as a separate header in the steps above).
`CanonicalizedAmzHeader`	The x-amz headers, which customize your S3 implementation. It's often left blank. We use `LF` in this example.
`CanonicalizedResource`	Your Amazon private bucket name. We use `"/test123"` in this example.

Following redirects to S3 objects and caching S3 responses

With custom VCL, Fastly can follow redirects to S3 objects and cache the s3 response as well as the 301 or 302 response separately.

> ❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Once the ability to upload custom VCL has been enabled, be sure to read our "How do I mix and match Fastly VCL with custom VCL? (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl)" instructions. It's important to include the entire VCL boilerplate if you do not intend to override the Fastly default settings.

To configure Fastly to follow redirects to S3 objects, insert the following VCL snippets in your custom VCL:

Within vcl_recv

```
sub vcl_recv {

  if (req.http.redir != "true") {
    set req.backend = Main_Origin;
  } else {
    set req.backend = s3_backend;
    set req.http.host = "s3.amazonaws.com";
  }

#FASTLY recv

  if (req.request != "HEAD" && req.request != "GET" && req.request != "FASTLYPURGE") {
    return(pass);
  }

  return(lookup);

}
```

Within vcl_deliver

```
sub vcl_deliver {

  if (resp.status == 302 || resp.status == 301) {
    set req.http.redir = "true";
    set req.url = regsub(resp.http.Location, "http://s3.amazonaws.com/(.*)$", "/\1");
    set req.http.Fastly-Force-Shield = "yes";
    restart;
  }

#FASTLY deliver

  return(deliver);
}
```

Be sure to set the `Main_Origin` and `s3_backend` to the actual name of your backends in the service to which you're applying these redirects. You can find the exact names by reviewing your VCL; simply click on the VCL button at the top of the page while viewing the service.

Once you added these VCL snippets to your custom VCL, upload the VCL file and then activate the new version of your service to apply the changes.

> This article describes an integration with a service provided by a third party. Please see our note on integrations (/guides/integrations/).

§ DigitalOcean Spaces (/guides/integrations/digitalocean-spaces)

DigitalOcean Spaces (https://www.digitalocean.com/products/storage/object-storage/) public and private Spaces can be used as origins with Fastly.

Using DigitalOcean Spaces as an origin

To make your DigitalOcean Spaces available through Fastly, follow the steps below.

Creating a new service

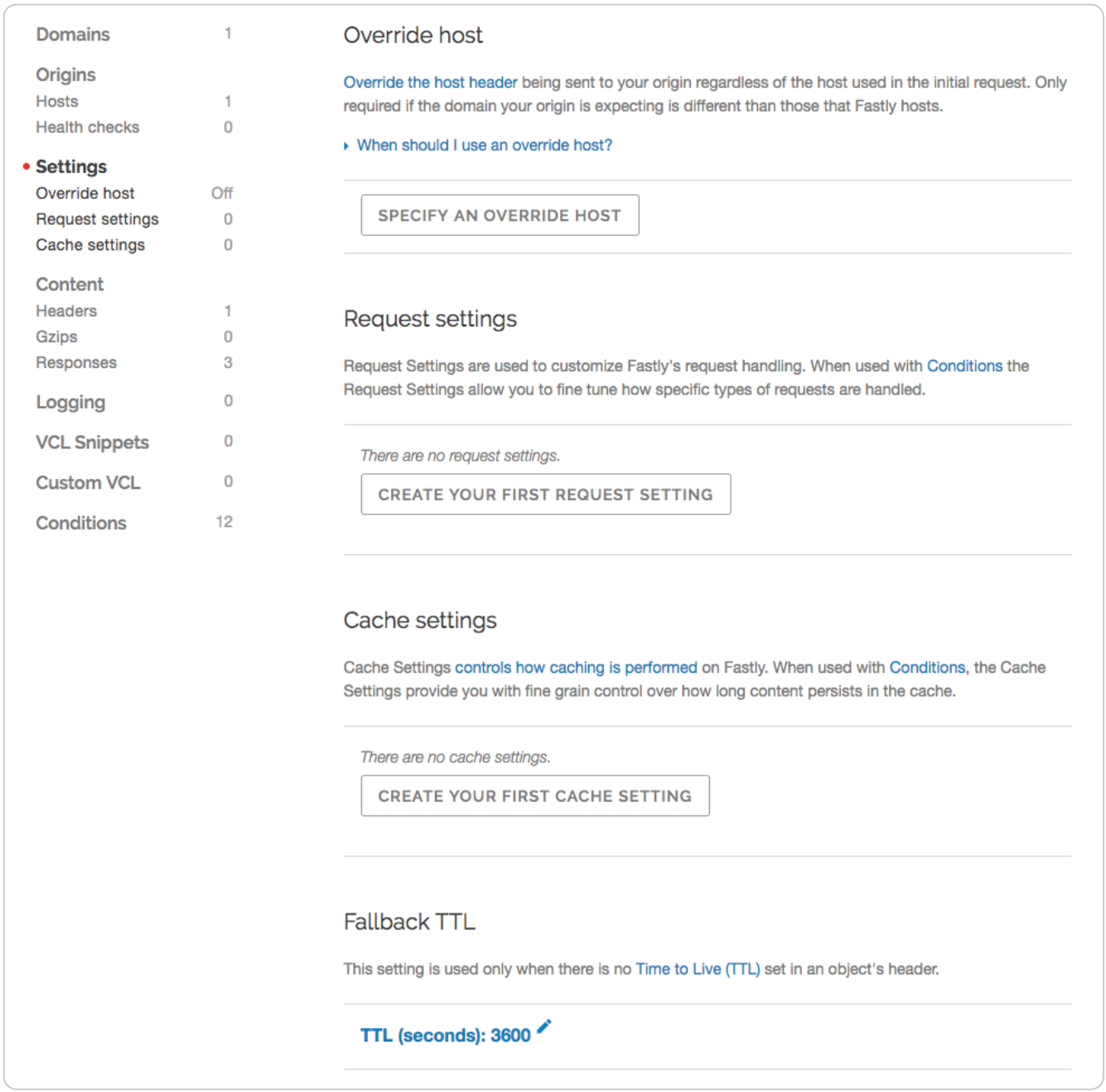
Follow the instructions for creating a new service (/guides/basic-setup/working-with-services#creating-a-new-service). You'll add specific details about your origin when you fill out the **Create a new service** fields:

- In the **Name** field, type any descriptive name for your service.
- In the **Domain** field, type the hostname you want to use as the URL (e.g., `cdn.example.com`).
- In the **Address** field, type `nyc3.digitaloceanspaces.com`. When DigitalOcean releases new regions, you can update the address here.
- In the **Transport Layer Security (TLS)** area, leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
- In the **Transport Layer Security (TLS)** area, type `nyc3.digitaloceanspaces.com` in the **Certificate hostname** field.

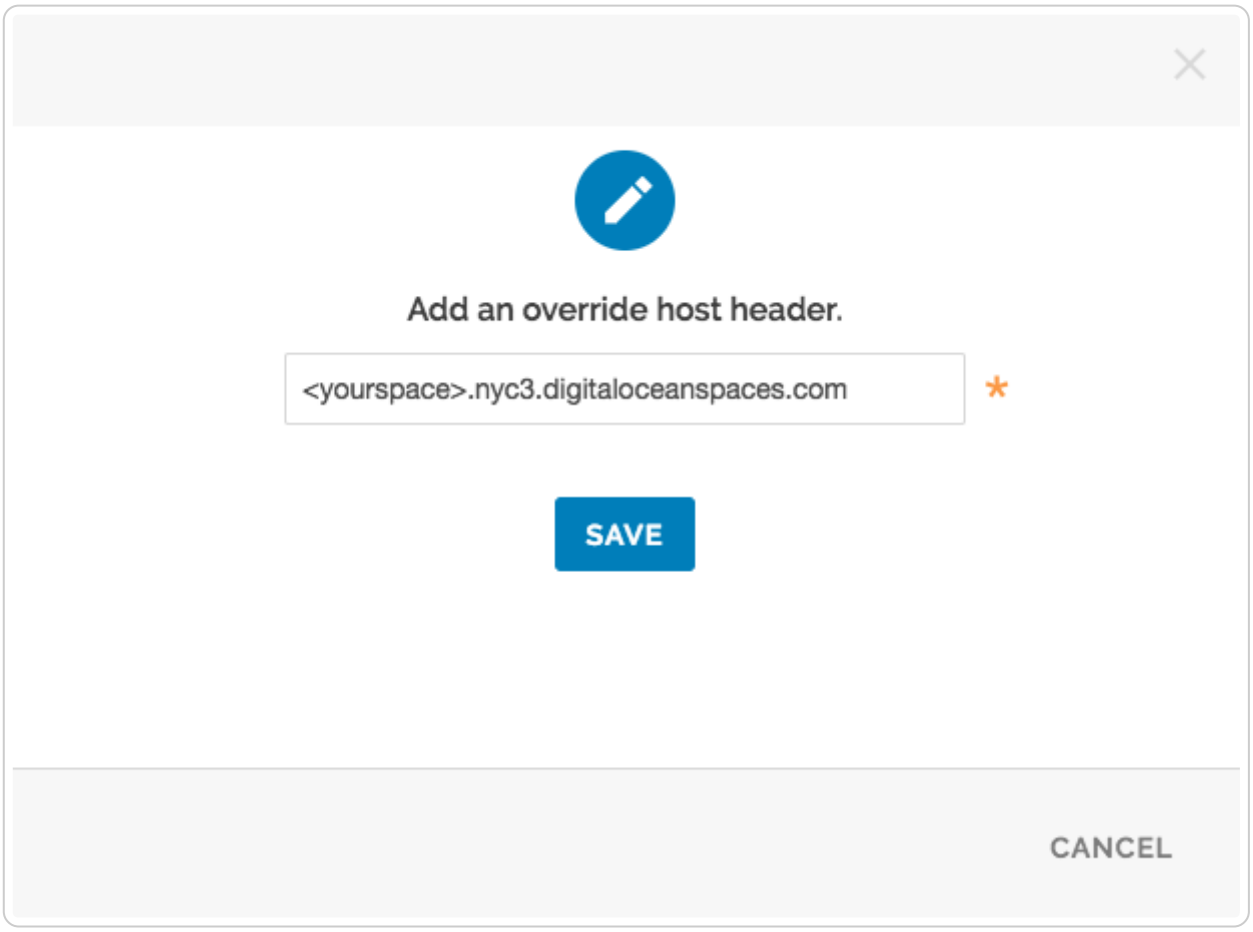
Setting the default host

Once the new service is created, set the default host to `<yourspace>.nyc3.digitaloceanspaces.com` by following the steps below:

1. From the service menu, select the appropriate service.
2. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
3. Click the **Settings** link. The Settings page appears.



4. Click the **Specify an Override Host** button. The Add an override host header window appears.



5. Type the hostname of your Space. For example, `<yourspace>.nyc3.digitaloceanspaces.com`.
6. Click the **Save** button.
7. Click the **Activate** button to deploy your configuration changes.

Testing your results

By default, we create DNS mapping called **yourdomain.global.prod.fastly.net**. In the example above, it would be `cdn.example.com.global.prod.fastly.net`. Create a DNS alias for the domain name you specified (e.g., CNAME `cdn.example.com` to `global-nossl.fastly.net`).

Fastly will cache any content without an explicit `Cache-Control` header for 1 hour. You can verify whether you are sending any cache headers using cURL. For example:

```
$ curl -I opscode-full-stack.nyc3.digitaloceanspaces.com

HTTP/1.1 200 OK
x-amz-id-2: ZpzRp7IWc6MJ8NtDEFGH12QBdk2CM1+RzVOngQbhMp2f2ZyalkFsZd4qPaLMkSlh
x-amz-request-id: ABV5032583242618
Date: Fri, 18 Mar 2012 17:15:38 GMT
Content-Type: application/xml
Transfer-Encoding: chunked
```

In this example, no cache control headers are set so default TTL will be applied.

Enhanced cache control

If you need more control over how different types of assets are cached (e.g., Javascript files, images), use the Amazon S3 configuration (/guides/tutorials/cache-control-tutorial) in our Cache Control tutorial as an example.

Using private DigitalOcean Spaces

To use a private DigitalOcean Space with Fastly, follow the instructions below.

Before you begin

Be sure you've already made your Spaces data available to Fastly by pointing to the right Space and setting your origin to port 443. This needs to be done before authenticating.

Be sure you've got the access key, secret key, and Space name on hand. The DigitalOcean Spaces Authorization header takes the following form:

```
Authorization: AWS `_AWSAccessKeyId_`:`_Signature_`
```

From the DigitalOcean website you will need the following information:

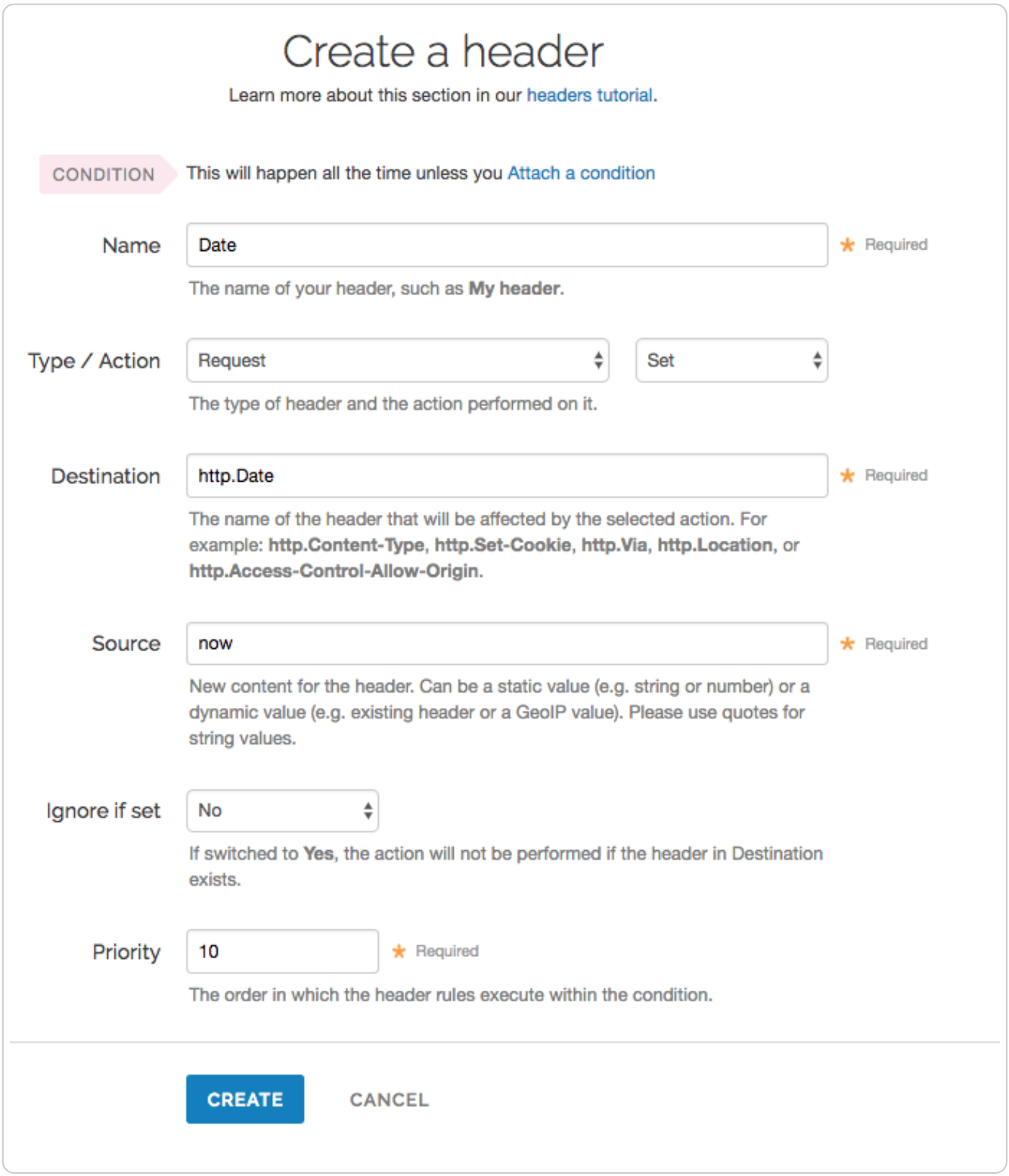
1. the **access key** and **secret key**
2. your **Space** name

Setting up Fastly to use a private DigitalOcean Space

In order to use a private DigitalOcean Space with Fastly, create two headers (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses), a Date header (for use with the authorization Signature) and an Authorization header.

Create a Date header

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Date`.
   - In the **Source** field, type `now`.

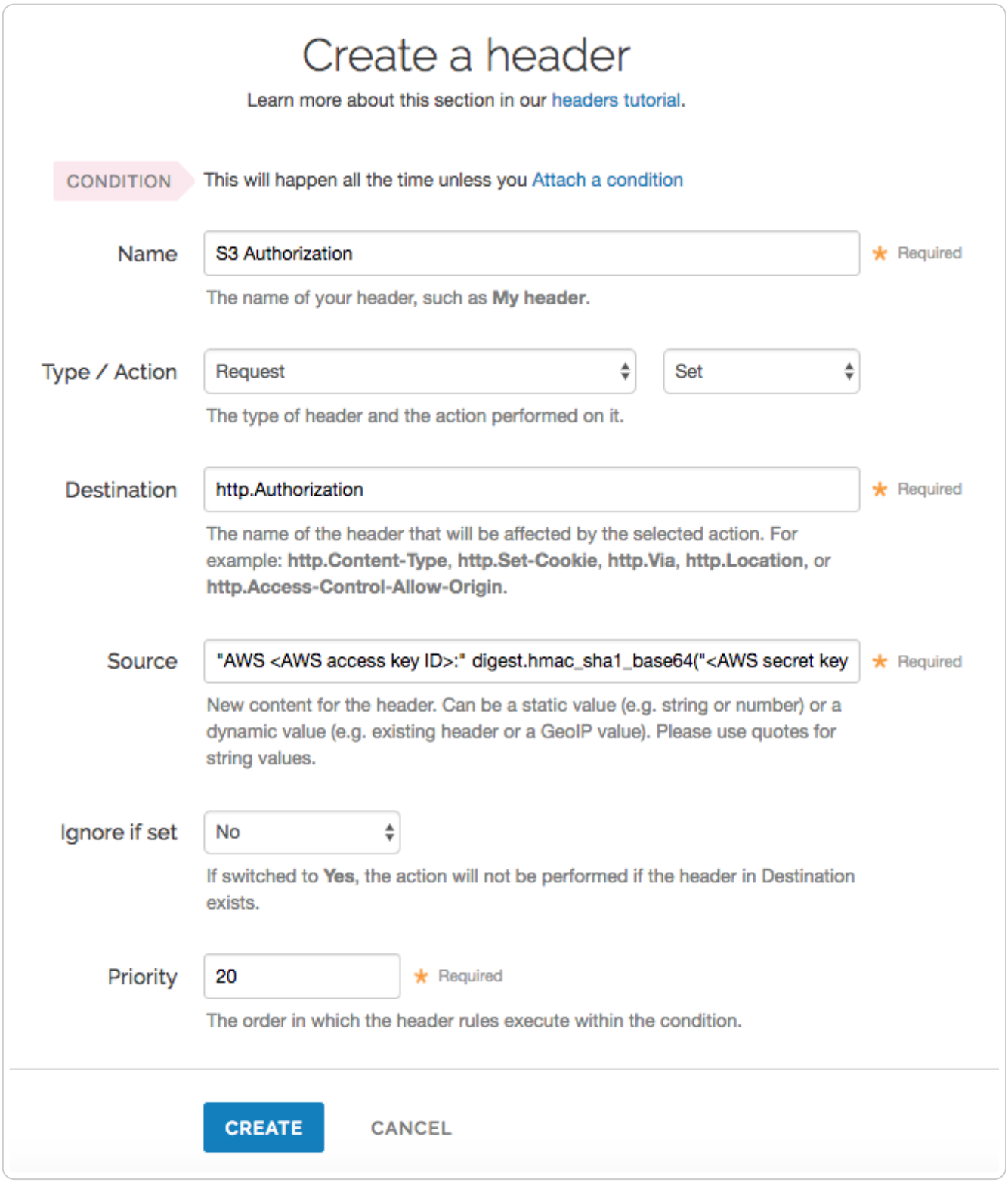
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button. A new Date header appears on the Content page. You will use this later within the Signature of the Authorization header.

Create an Authorization header

Next, create the Authorization header with the specifications listed below.

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Spaces Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.

- In the **Priority** field, type `20`.

3. In the **Source** field, type the header authorization information using the following format:

```
"AWS <DigitalOcean access key>:" digest.hmac_sha1_base64("<DigitalOcean secret key>", if(req.request == "HEAD", "GET", req.request) LF LF LF req.http.Date LF "/<Space name>" req.url.path)
```

   replacing `<DigitalOcean access key>`, `<DigitalOcean secret key ID>`, and `<Space name>` with the information you gathered before you began. For example:

```
"AWS JKCAUEFV2ONFFOFMSSLA:" digest.hmac_sha1_base64("P2WPSu68Bfl89j72vT+bXYZB7SjlOwhT4whqt27", if(req.request == "HEAD", "GET", req.request) LF LF LF req.http.Date LF "/test123" req.url.path)
```

4. Click the **Create** button. The new Authorization header appears on the Content page.

A detailed look at the Source field

So what's going on in the Source field of the Authorization header? Here's the basic format:

`AWS<Access Key><Signature Function><key><message>`

It tells us the following:

Element	Description
`AWS`	A constant placed before the access key. It's always AWS.
`access key`	The access key from your DigitalOcean account. We used `JKCAUEFV2ONFFOFMSSLA` in this example.
`signature function`	The algorithm used to validate the key and message of the signature. We used `digest.hmac_sha1_base64(<key>, <message>)` in this example.
`key`	The secret key from your DigitalOcean account. We used `P2WPSu68BfI89j72vT+bXYZB7SjIOwhT4whqt27` in this example.
`message`	The UTF-8 encoding of the StringToSign. See the table below for a break down of each portion of the message.

The message that's part of the Source field in the Authorization header takes on this basic format:

`<HTTP-verb></n><Content-MD5>/n<Content-Type></n><Date></n><CanonicalizedAmzHeader></n><CanonicalizedResource>`

It tells us the following:

Element	Description

Element	Description
`HTTP-verb`	The REST verb. We use `req.request` in this example. We rewrite HEAD to GET because Varnish does this internally before sending requests to origin.
`/n`	A newline indicator constant. It's always /n.
`Content-MD5`	The content-md5 header value, used as a message integrity check. It's often left blank. We use `LF` (line feed) in this example.
`Content-Type`	The content-type header value, used to specify the MIME-type. It's often left blank. We use `LF` in this example.
`Date`	The date and time stamp. We use `req.http.Date` (which we created first as a separate header in the steps above).
`CanonicalizedAmzHeader`	The x-amz headers, which customize your Spaces implementation. It's often left blank. We use `LF` in this example.
`CanonicalizedResource`	Your DigitalOcean Space name. We use `"/test123"` in this example.

Following redirects to Spaces objects and caching Spaces responses

With custom VCL, Fastly can follow redirects to Spaces objects and cache the Spaces response as well as the 301 or 302 response separately.

> ❗ **IMPORTANT:** The ability to upload custom VCL code (/guides/vcl/uploading-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Once the ability to upload custom VCL has been enabled, be sure to read our instructions about mixing and matching Fastly VCL with custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl). It's important to include the entire VCL boilerplate if you do not intend to override the Fastly default settings.

To configure Fastly to follow redirects to Spaces objects, insert the following VCL in your custom VCL:

Within vcl_recv

```
sub vcl_recv {

  if (req.http.redir != "true") {
    set req.backend = Main_Origin;
  } else {
    set req.backend = spaces_backend;
    set req.http.host = "nyc3.digitaloceanspaces.com";
  }

#FASTLY recv

  if (req.request != "HEAD" && req.request != "GET" && req.request != "FASTLYPURGE") {
    return(pass);
  }

  return(lookup);

}
```

Within vcl_deliver

```
sub vcl_deliver {

  if (resp.status == 302 || resp.status == 301) {
    set req.http.redir = "true";
    set req.url = regsub(resp.http.Location, "http://nyc3.digitaloceanspaces.com/(.*)$", "/\1"
);
    set req.http.Fastly-Force-Shield = "yes";
    restart;
  }

#FASTLY deliver

  return(deliver);
}
```

Be sure to set the `Main_Origin` and `spaces_backend` to the actual name of your backends in the service to which you're applying these redirects. You can find the exact names by reviewing your VCL. Simply click on the VCL button at the top of the page while viewing the service.

Once you added these VCL snippets to your custom VCL, upload the VCL file and then activate the new version of your service to apply the changes.

> This article describes an integration with a service provided by a third party. Please see our note on integrations (/guides/integrations/).

§ Google Cloud Storage (/guides/integrations/google-cloud-storage)

Google Cloud Storage (https://cloud.google.com/storage/) (GCS) can be used as an origin server with your Fastly services once you set up and configure your GCS account and link it to a Fastly service. It can also be configured to use private content. This speeds up your content delivery and reduces your origin's workload and response times with the dedicated links between Google and Fastly's POPs.

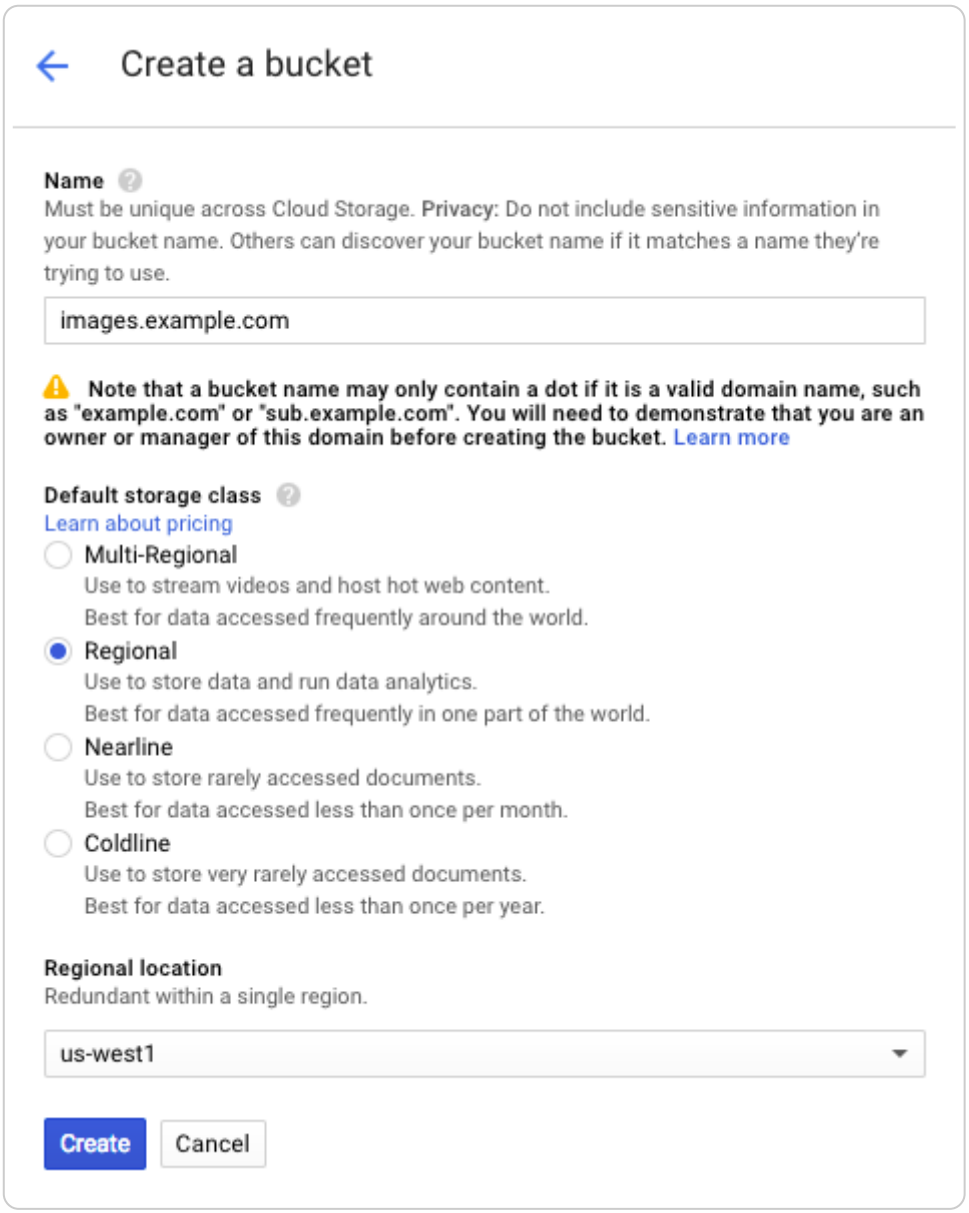
> ★ **TIP:** Google offers a Cloud Accelerator (/guides/about-fastly-services/about-fastlys-cloud-accelerator) integration discount that applies to any Google Cloud Platform product. If you're a Fastly customer and would like to take advantage of this discount, email salesgcp@fastly.com (mailto:salesgcp@fastly.com).

Using GCS as an origin server

To make your GCS data available through Fastly, follow the steps below.

Setting up and configuring your GCS account

1. Sign up for Google Cloud Storage (https://cloud.google.com/storage/docs/signup).
2. Create a bucket (https://cloud.google.com/storage/docs/getting-started-console) to store your origin's data. The Create a bucket window appears.



3. Use Google's Search Console (https://www.google.com/webmasters/verification/) to verify ownership of your domain name, if you have not already done so. See the instructions on Google's website (https://cloud.google.com/storage/docs/domain-name-verification).
4. Fill out the **Create a bucket** fields as follows:
   - In the **Name** field, type your domain name (e.g., `example.com` or `images.example.com`) to create a domain-named bucket (https://cloud.google.com/storage/docs/domain-name-verification). Remember the name you type. You'll need it to connect your GCS bucket to your Fastly service.
   - In the **Default storage class** area, select **Regional**.
   - From the **Regional location** menu, select a location to store your content. Most customers select a region close to the interconnect location they specify for shielding (/guides/performance-tuning/shielding).
5. Click the **Create** button.

You should now add files to your bucket and make them externally accessible by selecting the **Public link** checkbox next to each of the files.

Adding your GCS bucket as an origin server

To add your GCS bucket as an origin server, follow the instructions for connecting to origins (/guides/basic-configuration/connecting-to-origins). You'll add specific details about your origin server when you fill out the **Create a host** fields:

- In the **Name** field, type the name of your server (for example, `Google Cloud Storage`).
- In the **Address** field, type `storage.googleapis.com`.
- In the **Transport Layer Security (TLS)** area, leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
- In the **Transport Layer Security (TLS)** area, type `storage.googleapis.com` in the **Certificate hostname** field.
- From the **Shielding** menu, select an interconnect location from the list of shielding locations.

Interconnect locations

Interconnect locations allow you to establish direct links with Google's network edge when you choose your shielding location. By selecting one of the locations listed below, you will be eligible to receive discounted pricing (https://cloud.google.com/interconnect/cdn-interconnect#pricing) from Google CDN Interconnect for traffic traveling from Google Cloud Platform to Fastly's network. Most customers select the interconnect closest to their GCS bucket's region.

Interconnects exist in the following locations within North America:

- Ashburn (DCA)
- Ashburn (IAD)
- Atlanta (ATL)
- Chicago (MDW)
- Dallas (DFW)
- Los Angeles (LAX)
- New York (JFK)
- Seattle (SEA)
- San Jose (SJC)
- Toronto (YYZ)

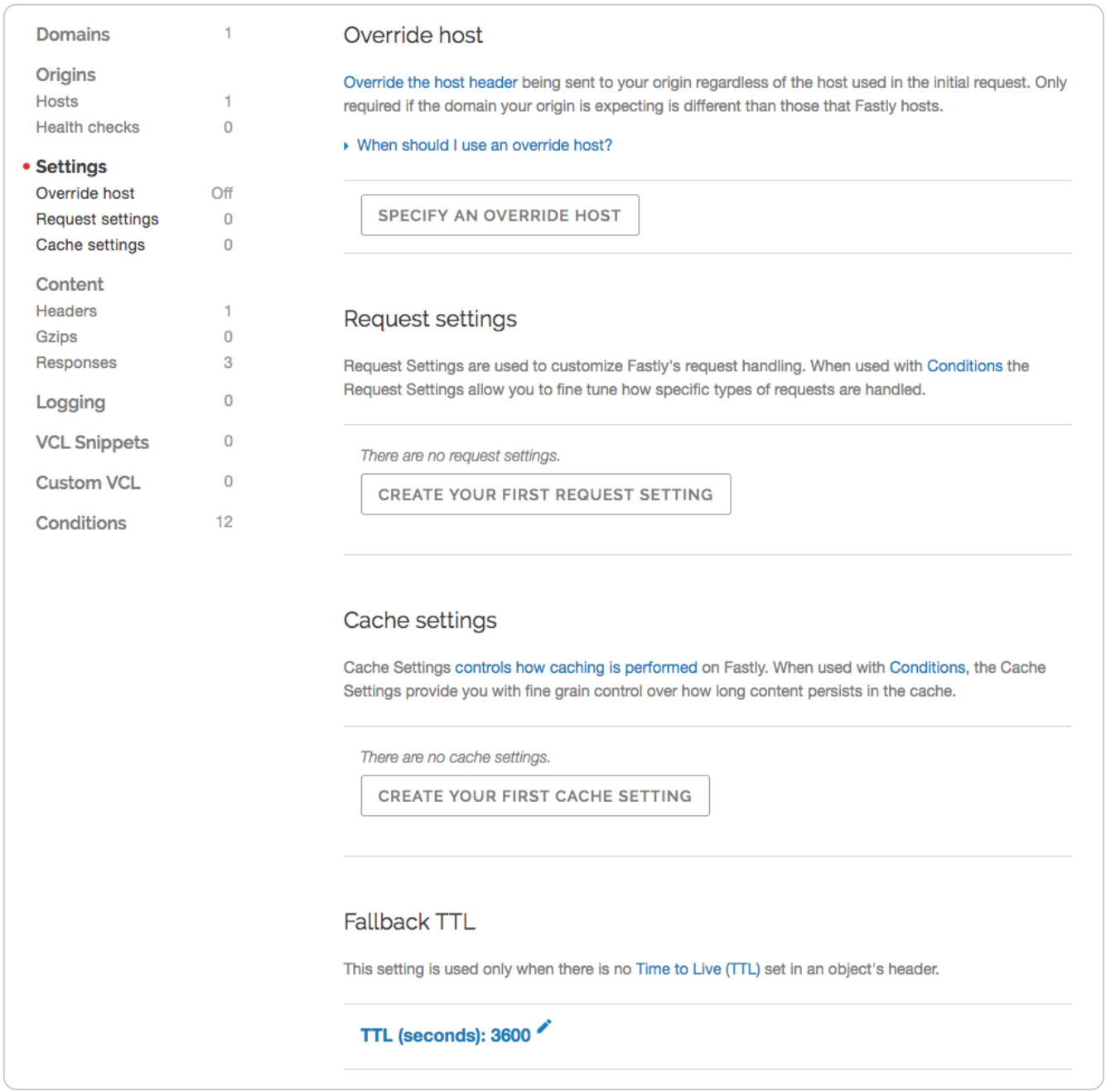
Interconnects outside of North America exist in:

- Amsterdam (AMS)
- Frankfurt (FRA)
- Frankfurt (HHN)
- Hong Kong (HKG)
- London (LCY)
- London (LHR)
- Madrid (MAD)
- Paris (CDG)
- Singapore (SIN)
- Stockholm (BMA)
- Tokyo (NRT)

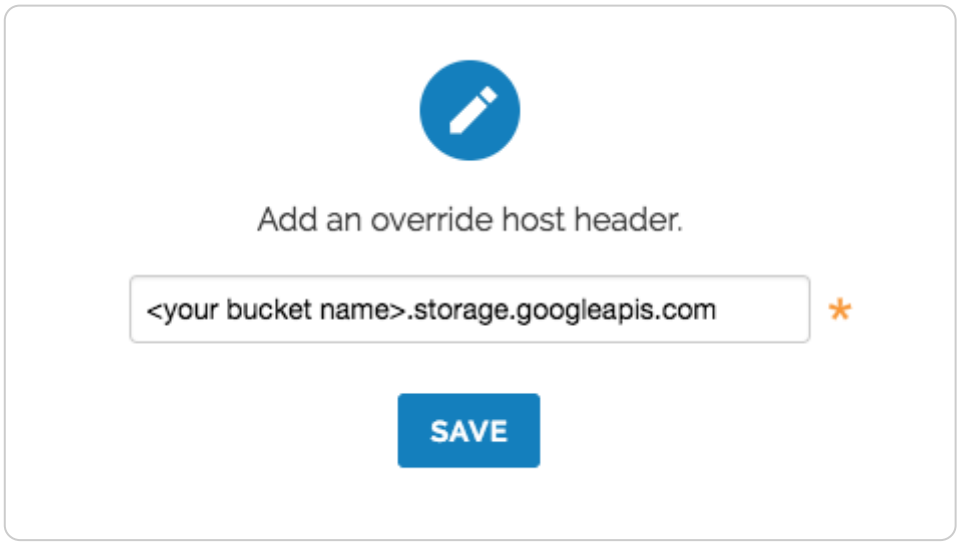
Review our caveats of shielding (/guides/performance-tuning/shielding#caveats-of-shielding) and select an interconnect accordingly.

Setting the default host for your service to your GCS bucket

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.



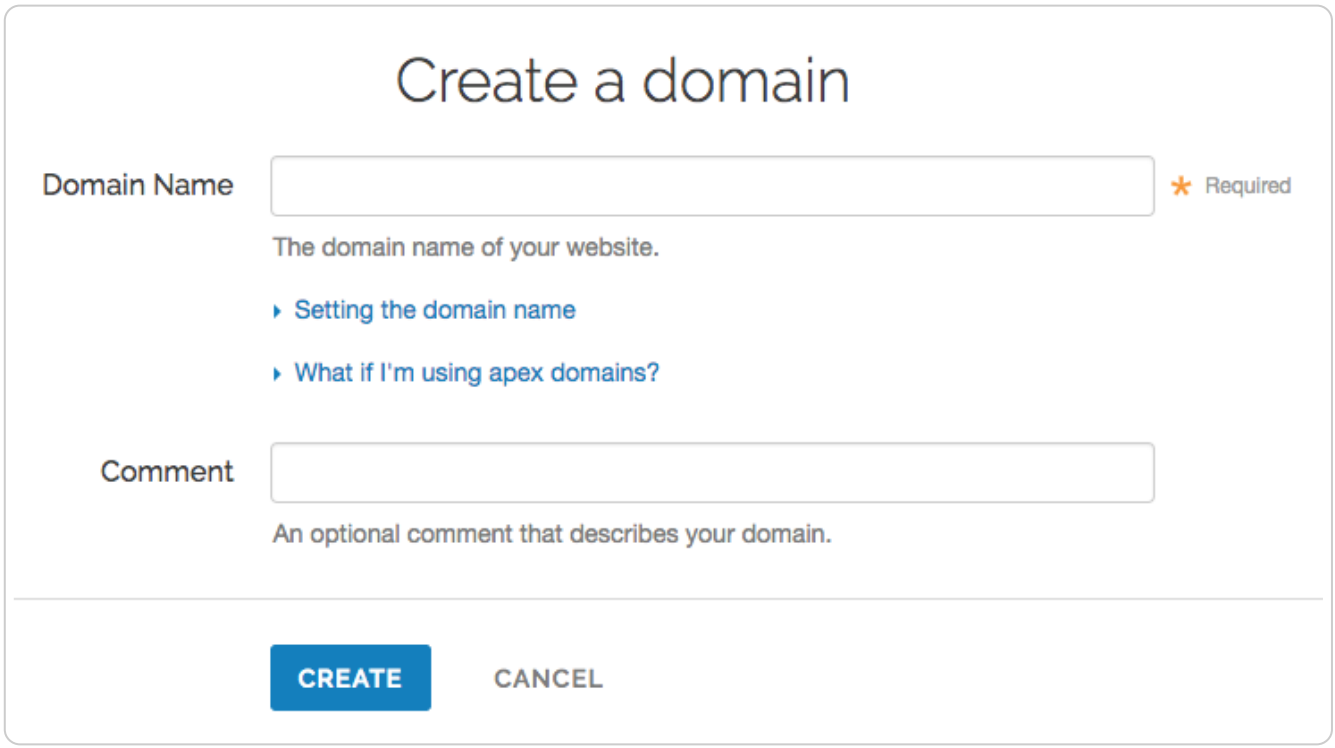
5. Click the **Specify an override host** button. The Add an override host header window appears.



6. Type the name of the override host for this service. The name you type should match the name of the bucket you created in your GCS account and will take the format `<your bucket name>.storage.googleapis.com`. For example, if your bucket name is `test123`, your override hostname would be `test123.storage.googleapis.com`.
7. Click the **Save** button. A new override host appears on the Settings page.

Creating domains for GCS

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Create domain** button. The Create a domain page appears.



5. In the **Domain Name** field, type the name users will type in their browsers to access your site.
6. In the **Comment** field, optionally type a comment that describes your domain.
7. Click the **Create** button. A new domain appears on the Domains page.
8. Because GCS responds to different hostnames than your Fastly service, click the **Create domain** button to create a second domain.
9. In the **Domain Name** field of the second domain you create, type the same value as the default host you created earlier (e.g., `<your bucket name>.storage.googleapis.com`) and click the **Create** button. A second new domain appears on the Domains page. Shielding POPs need this additional domain so they can route requests correctly. (See Caveats of shielding (/guides/performance-tuning/shielding#caveats-of-shielding) for more information.)
10. Click the **Activate** button to deploy your configuration changes.
11. Add a CNAME DNS record (/guides/basic-setup/adding-cname-records) for the domain if you haven't already done so.

You can use `http://<domain>.global.prod.fastly.net/<filename>` to access the files you uploaded.

Setting the Cache-Control header for your GCS bucket

GCS performs its own caching, which may complicate efforts to purge cache. To avoid potential problems, we recommend using the gsutil (https://cloud.google.com/storage/docs/gsutil_install) command line utility to set the Cache-Control header for one or more files in your GCS bucket:

```
gsutil setmeta -h "Cache-Control: max-age=0, s-maxage=86400" gs://<bucket>/*.html
```

Replace `<bucket>` in the example above with your GCS bucket's name. Note that `max-age` should instruct GCS to cache your content for zero seconds, and Fastly to cache your content for one day. See Google's setmeta docs (https://cloud.google.com/storage/docs/gsutil/commands/setmeta) for more information.

Changing the default TTL for your GCS bucket

If you want to change the default TTL for your GCS bucket, if at all, keep the following in mind:

- Your GCS account controls the default TTL for your GCS content. GCS currently sets the default TTL to 3600 seconds. Changing the default TTL will not override the default setting in your GCS account.
- To override the default TTL set by GCS from within the Fastly web interface, create a new cache setting (/guides/performance-tuning/controlling-caching) and enter the TTL there.
- To override the default TTL in GCS, download the gsutil tool (https://cloud.google.com/storage/docs/gsutil_install) and then change the cache-control headers (https://cloud.google.com/storage/docs/gsutil/addlhelp/WorkingWithObjectMetadata#cache-control) to delete the default TTL or change it to an appropriate setting.

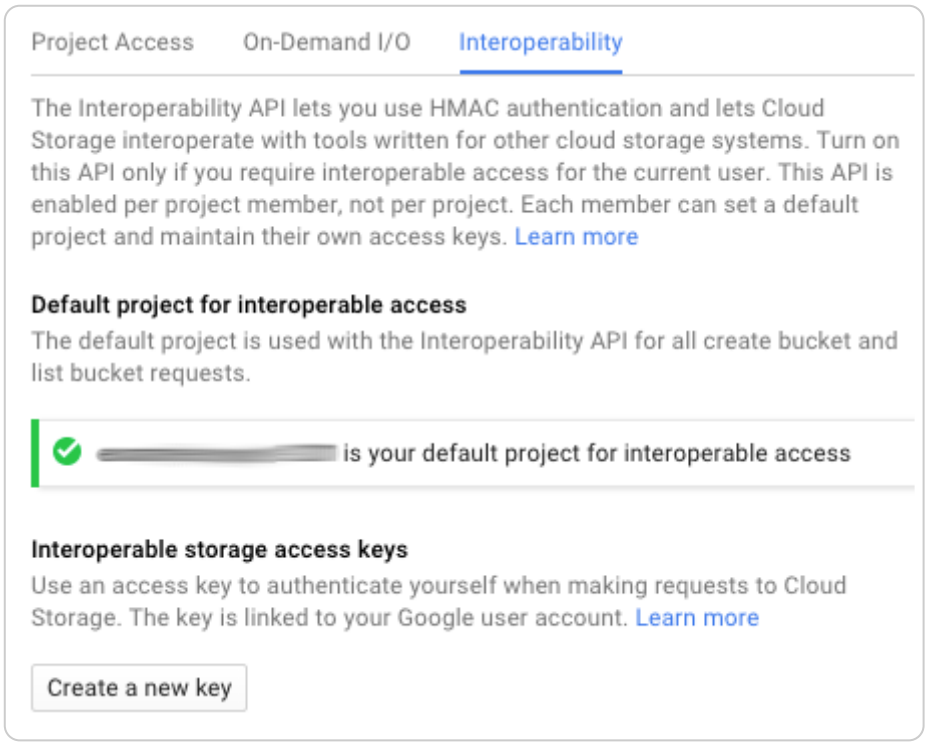
Using GCS with private objects

To use Fastly with GCS private objects, be sure you've already made your GCS data available to Fastly by pointing to the right GCS bucket, then follow the steps below.

Setting up interoperable access

By default, GCS authenticates requests using OAuth2, which Fastly does not support. To access private objects on GCS, your project must have HMAC authentication (https://cloud.google.com/storage/docs/gsutil/addlhelp/CredentialTypesSupportingVariousUseCas credential-types) enabled and interoperable storage access keys (an "Access Key" and "Secret" pair) created. Do this by following the steps below.

1. Open the Google Cloud Platform console and select the appropriate project.
2. Click **Settings**. The Settings appear with the Project Access controls highlighted.
3. Click the **Interoperability** tab. The Interoperability API access controls appear.
4. If you have not set up interoperability before, click **Enable interoperability access**.
5. Click **Make `<PROJECT-ID>` your default project** for interoperable access. If that project already serves as the default project, that information appears instead.



6. Click **Create a new key**. An access key and secret code appear.



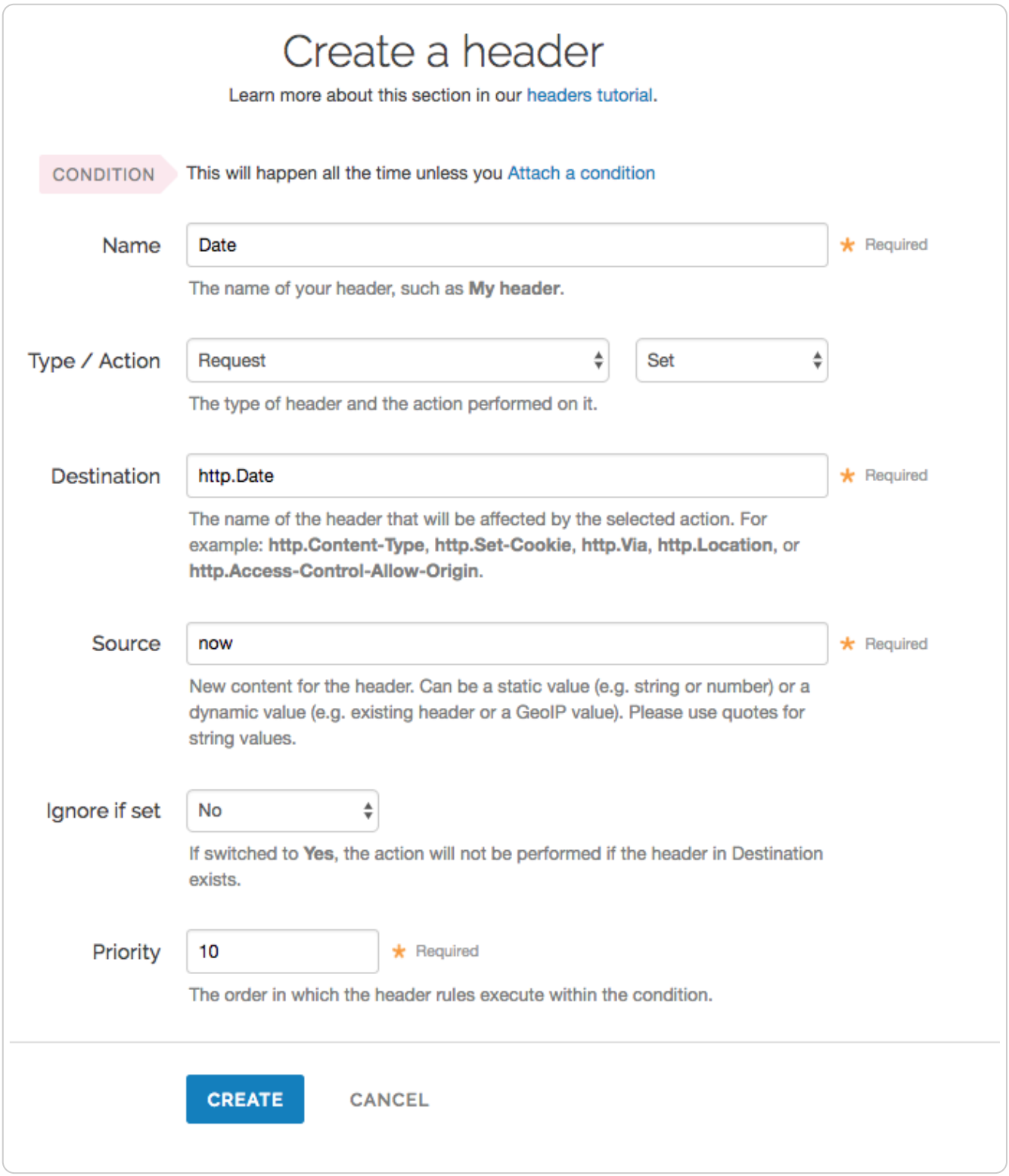
7. Save the access key and secret code that appear. You'll need these later when you're creating an authorization header.

Setting up Fastly to use GCS private content

To use GCS private content with Fastly, create two headers (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses), a Date header (required Authorization Signature) and an Authorization header.

Creating a Date header

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a new header page appears.



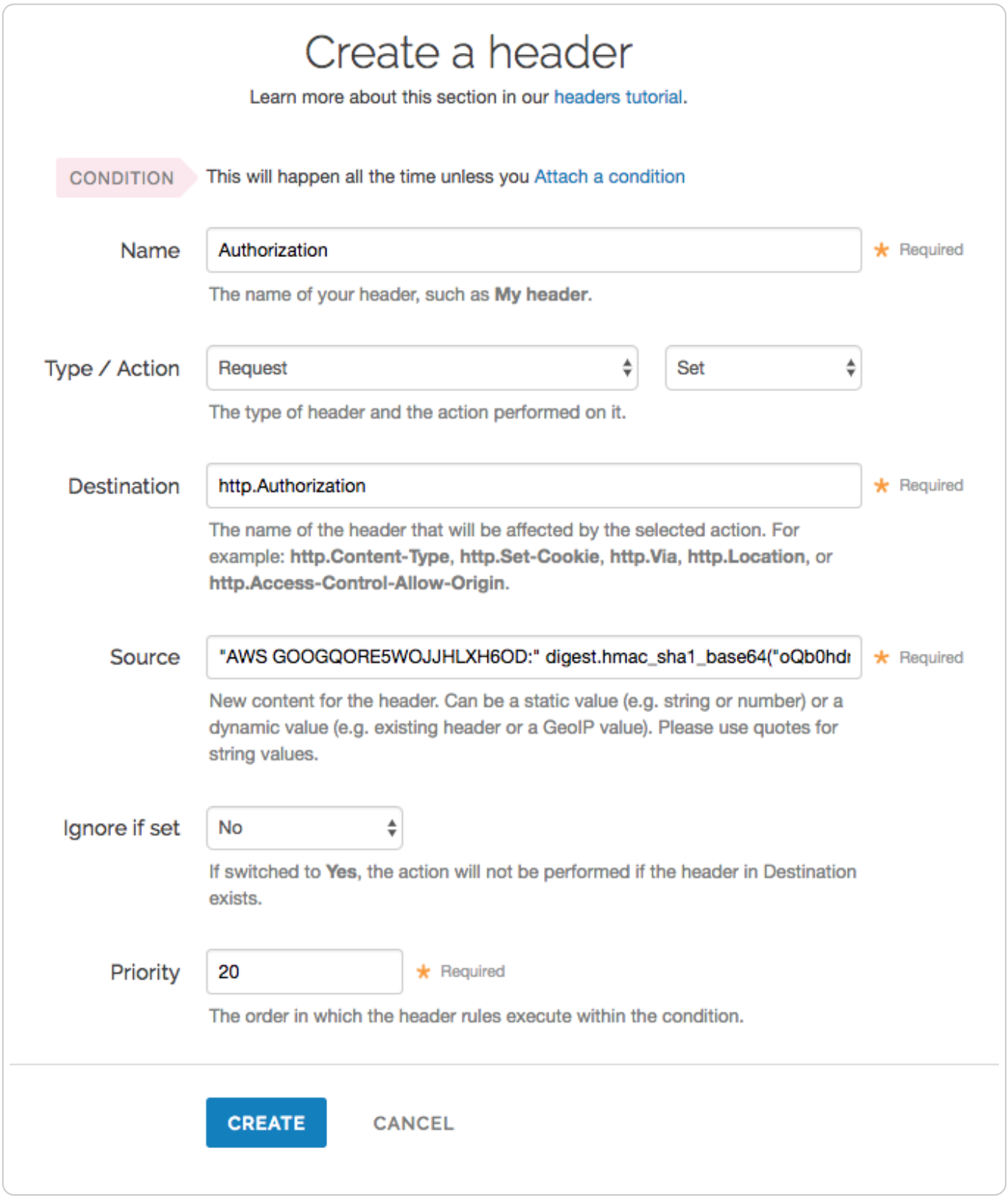
6. Fill out the **Create a new header** fields as follows:
   - In the **Name** field, type `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Date`.
   - In the **Source** field, type `now`.

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

7. Click the **Create** button. A new Date header appears on the Content page. You will use this later within the Signature of the Authorization header.

Creating an Authorization header

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.

- In the **Priority** field, type `20`.

3. In the **Source** field, type the header authorization information using the following format:

```
  "AWS <access key>:" digest.hmac_sha1_base64("<GCS secret>", req.request LF LF LF req.htt
p.Date LF "/<GCS bucket name>" req.url.path)
```

   replacing `<access key>`, `<GCS secret>`, and `<GCS bucket name>` with the information you gathered before you began. For example:

```
  "AWS GOOGQORE5WOJJHLXH6OD:" digest.hmac_sha1_base64("oQb0hdmaxFOc5UmC6F833Cde0+ghRSgsr7C
CnX62", req.request LF LF LF req.http.Date LF "/test123" req.url.path)
```

4. Click the **Create** button. A new Authorization header appears on the Content page.
5. Click the **Activate** button to deploy your configuration changes.

A detailed look at the Source field

So what's going on in the Source field of the Authorization header? Here's the basic format:

`AWS<access key><signature function><key><message>`

It tells us the following:

Element	Description
`AWS`	A constant placed before the access key. It's always AWS.
`access key`	The access key ID from your GCS developer's account. We used `GOOGQORE5WOJJHLXH6OD` in this example.
`signature function`	The algorithm used to validate the key and message of the signature. We used `digest.hmac_sha1_base64(<key>, <message>)` in this example.
`key`	The secret key ID from your GCS developer's account. We used `oQb0hdmaxFOc5UmC6F833Cde0+ghRSgsr7CCnX62` in this example.
`message`	The UTF-8 encoding of the StringToSign. See the table below for a break down of each portion of the message.

The message that's part of the Source field in the Authorization header takes on this basic format:

`<HTTP-verb><\n><Content-MD5>\n<Content-Type><\n><Date><\n><CanonicalExtensionHeaders><\n><CanonicalizedResource>`

It tells us the following:

Element	Description
`HTTP-verb`	The REST verb. We use `req.request` in this example.

Element	Description
`\n`	A newline indicator constant. It's always \n.
`Content-MD5`	The content-md5 header value, used as a message integrity check. It's often left blank. We use `LF` (line feed) in this example.
`Content-Type`	The content-type header value, used to specify the MIME-type. It's often left blank. We use `LF` in this example.
`Date`	The date and time stamp. We use `req.http.Date` (which we created first as a separate header in the steps above).
`CanonicalExtensionHeaders`	The x-amz- or x-goog- headers, which customize your GCS implementation. It's often left blank. We use `LF` in this example.
`CanonicalizedResource`	Your GCS resource path name. We're concatenating GCS bucket name `"/test123"` with object path `req.url.path` in this example.

This article describes an integration with a service provided by a third party. Please see our note on integrations (/guides/integrations/).

§ Google Compute Engine (/guides/integrations/google-compute-engine)

Google Compute Engine (GCE) lets you create and run a virtual machine (VM) on the Google infrastructure. The VM can be used as an origin server with your Fastly service once you set up and configure your VM instance and link your instance to a Fastly service.

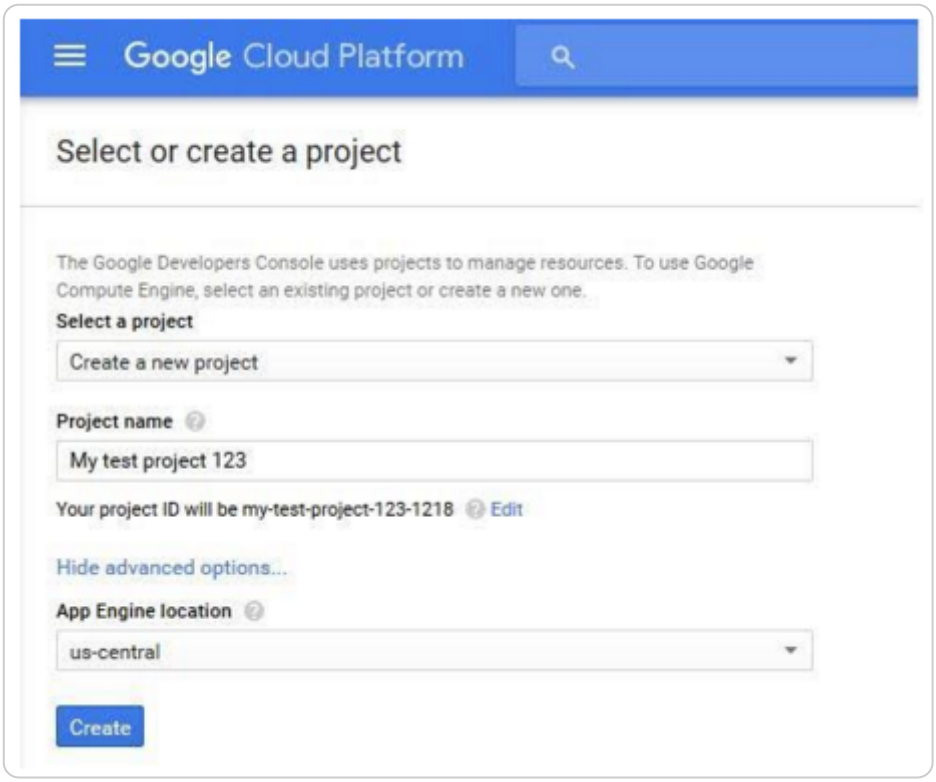
★ **TIP:** Google offers a Cloud Accelerator (/guides/about-fastly-services/about-fastlys-cloud-accelerator) integration discount that applies to any Google Cloud Platform product. If you're a Fastly customer and would like to take advantage of this discount, email salesgcp@fastly.com (mailto:salesgcp@fastly.com).

Creating and setting up your GCE instance

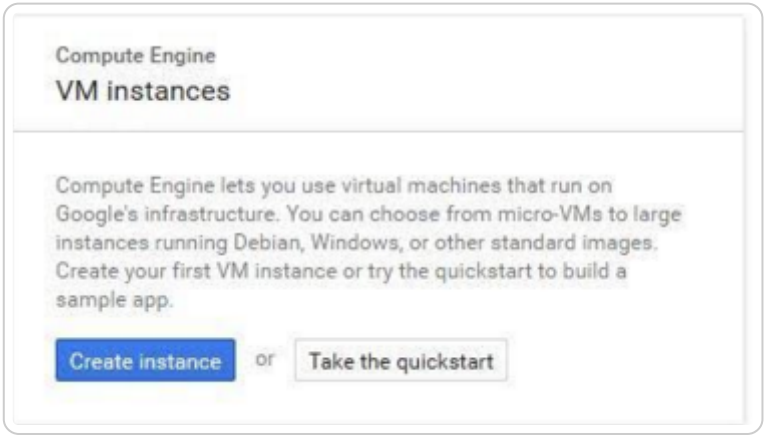
1. Sign up for Google Compute Engine (https://cloud.google.com/compute/docs/) and start the basic set up. If you are already signed up and at your dashboard, click the **Get started** link in

the Try Compute Engine area.

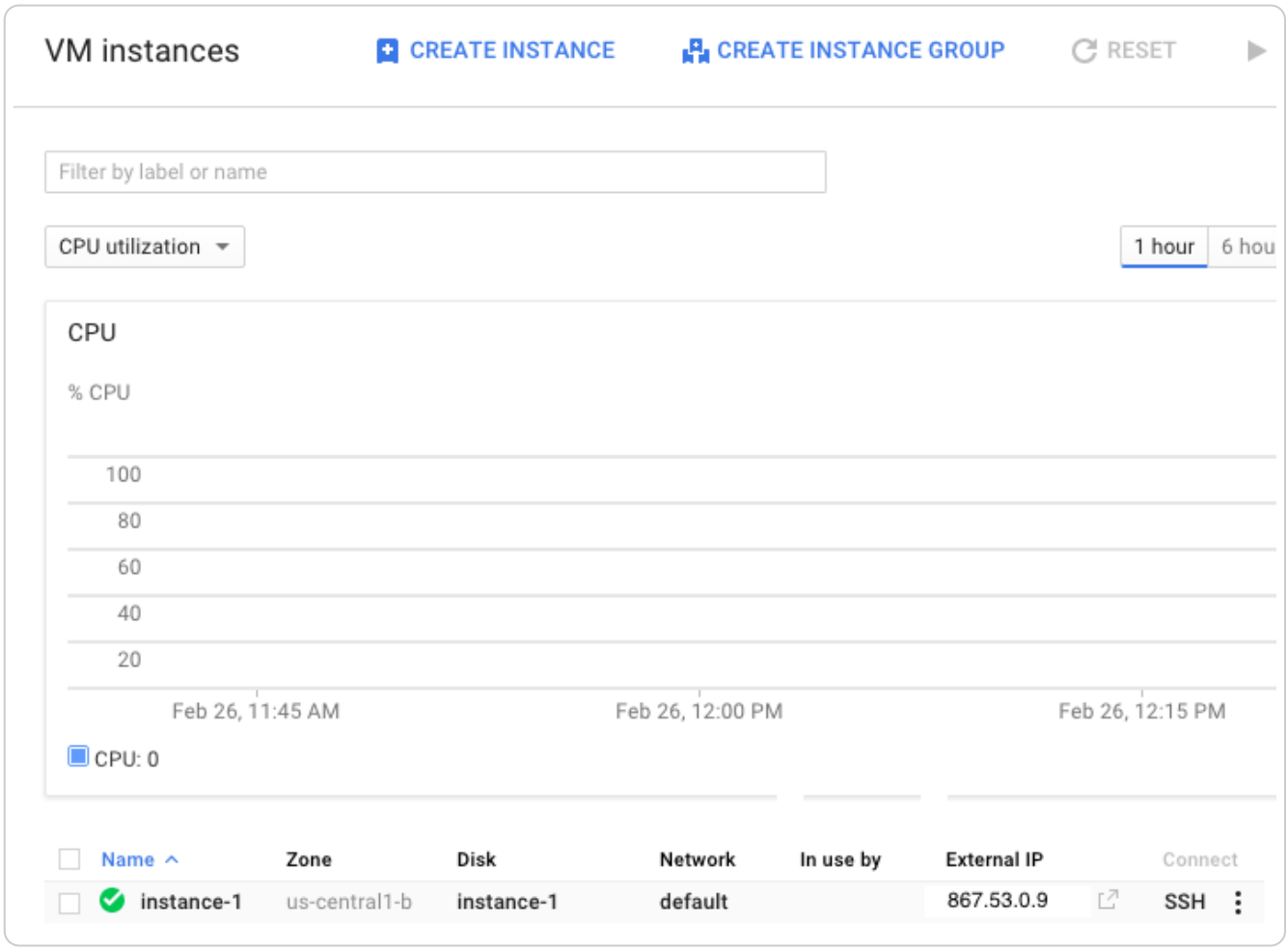
2. Create or select a project to hold your origin's data.



3. Click **Create instance** to set up your VM. You can set up your instance using either Windows or Linux.



4. Fill in the necessary fields and click **Create**. When making your firewall selection, select either **Allow HTTPS traffic** (port 443) or **Allow HTTP traffic** (port 80); you will use one of those ports when you create your new origin in your Fastly service. If you select HTTPS traffic, you need to configure the VM to respond on port 443 with a valid TLS certificate.



5. Make note of the following information for when you create your new origin in your Fastly service:
   - The instance's IP address (located in the External IP column at the bottom of the page). You'll use this in the **Address** field when you create your new origin.
   - The zone you are using (located in the Zone column at the bottom of the page). You'll use this to guide your selection of an appropriate shielding location for your origin.

Creating a new origin in your Fastly service for your GCE account

Link your GCE account to a Fastly service following the steps below.

1. Log in to the Fastly web interface and click the **Configure** link.
2. Create a new service (/guides/basic-setup/working-with-services) if you don't already have one set up.
3. From the service menu, select the appropriate service.
4. Follow the instructions for connecting to origins (/guides/basic-configuration/connecting-to-origins). You'll add specific details about your origin server when you fill out the **Create a host** fields:

- In the **Name** field, type the name of your server (for example, `Google Compute Engine`).
- In the **Address** field, type the IP address of your server. This should match the port that you selected in the GCE interface.
- From the **Shielding** menu, select an interconnect location from the list of shielding locations.

Interconnect locations

Interconnect locations allow you to establish direct links with Google's network edge when you choose your shielding location. By selecting one of the locations listed below, you will be eligible to receive discounted pricing (https://cloud.google.com/interconnect/cdn-interconnect#pricing) from Google CDN Interconnect for traffic traveling from Google Cloud Platform to Fastly's network. Most customers select the interconnect closest to their origin.

Interconnects exist in the following locations within North America:

- Ashburn (DCA)
- Ashburn (IAD)
- Atlanta (ATL)
- Chicago (MDW)
- Dallas (DFW)
- Los Angeles (LAX)
- New York (JFK)
- Seattle (SEA)
- San Jose (SJC)
- Toronto (YYZ)

Interconnects outside of North America exist in:

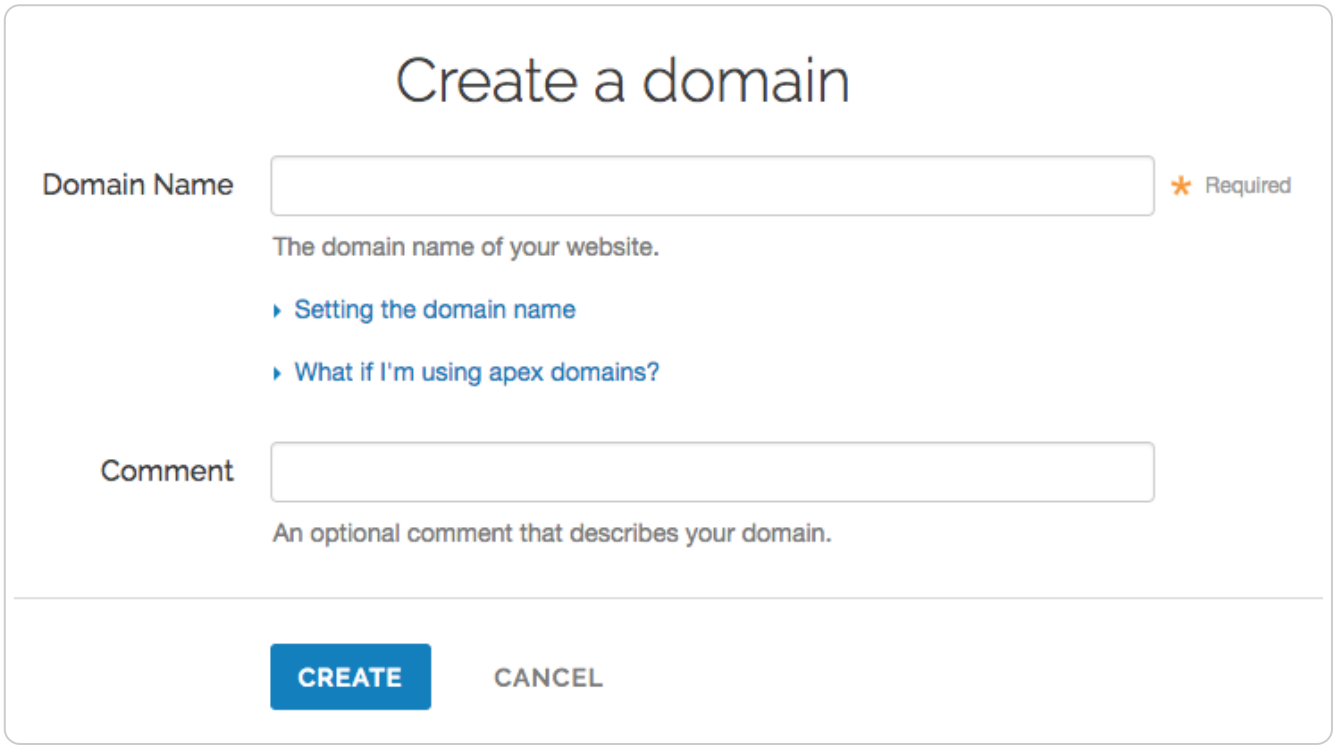
- Amsterdam (AMS)
- Frankfurt (FRA)
- Frankfurt (HHN)
- Hong Kong (HKG)
- London (LCY)
- London (LHR)
- Madrid (MAD)
- Paris (CDG)

- Singapore (SIN)
- Stockholm (BMA)
- Tokyo (NRT)

Review our caveats of shielding (/guides/performance-tuning/shielding#caveats-of-shielding) and select an interconnect accordingly.

Creating new domains for GCE to respond to

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Create domain** button. The Create a domain page appears.



5. In the **Domain Name** field, type the name that users will type in their browsers to access your site.
6. In the **Comment** field, optionally type a comment that describes your domain.
7. Click the **Create** button. The new domain appears on the Domains page.
8. Click the **Activate** button to deploy your configuration changes.

Creating a CNAME record

You can now test your configuration (/guides/basic-configuration/testing-setup-before-changing-domains). In the example above, your domain would appear as `www.example.com.global-nossl.fastly.net`. After you test and you're satisfied with the results, create a CNAME record (/guides/basic-setup/adding-cname-records) for your domain (e.g., `www.example.com`) pointing to `global-nossl.fastly.net`.

This article describes an integration with a service provided by a third party. Please see our note on integrations (/guides/integrations/).

§ PerimeterX Bot Defender (/guides/integrations/perimeterx-bot-defender)

Fastly provides direct integration between PerimeterX Bot Defender (https://www.perimeterx.com/products/bot-defender/?utm_source=partner&utm_medium=fastly&utm_campaign=botdefender) and Fastly edge servers. By placing a snippet of JavaScript (or HTML5) on your site and custom VCL (/guides/vcl/guide-to-vcl) directly into your Fastly service configuration, this integration allows you to gather behavioral data and statistics that may help you do things like detect invalid traffic and mitigate automated web attacks.

IMPORTANT: This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

How to get started

Integration with PerimeterX Bot Defender requires an account with PerimeterX. Once you have this account set up, contact your Fastly account manager or email sales@fastly.com (mailto:sales@fastly.com) to begin the integration process with Fastly. We'll work with you to configure your service to include the required code to enforce bot mitigation policies.

This article describes an integration with a service provided by a third party. Please see our note on integrations (/guides/integrations/).

§ Zencoder (/guides/integrations/zencoder)

Zencoder provides HLS output that Fastly can cache and deliver. To start, you'll need to create a Zencoder job that outputs live HLS to S3. Zencoder provides documentation on this setup process (https://app.zencoder.com/docs/guides/getting-started/live-s3-quickstart).

Next, configure a new Fastly service using the S3 bucket that you are using with Zencoder as an origin. You can refer to our documentation on setting up S3 as an origin (/guides/integrations/amazon-s3).

Once this is complete, you can serve your HLS streams through Fastly as you would any other content.

> This article describes an integration with a service provided by a third party. Please see our note on integrations (/guides/integrations/).

---

§ About Dynamic Servers (/guides/dynamic-servers/about-dynamic-servers)

Fastly's Load Balancer allows you to create pools of origin servers that you dynamically manage using Fastly's Dynamic Servers feature to distribute and direct incoming requests. The benefits include:

- support for any infrastructure deployments including any type of server instances and one or more datacenters, regions, or cloud providers
- high availability of web applications when using health checks
- compatibility with TLS termination, HTTP/2, and IPv6
- server stickiness without requiring a cookie-based approach
- implement any number of request routing rules/conditions to select a pool of origin servers

To set up Dynamic Servers, you attach a pool to a service (/guides/dynamic-servers/creating-and-using-pools-with-dynamic-servers), then add versionless origin servers that are stored separately from your VCL configuration. You can use the Fastly API (/api/dynamicservers) to programmatically add, remove, and update pools and origin servers.

> ❗ **IMPORTANT:** This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

How Dynamic Servers work

Like Edge Dictionaries (/guides/edge-dictionaries/about-edge-dictionaries) and Edge ACLs (/guides/access-control-lists/about-edge-acls), Dynamic Servers have two major components: the pool and origin servers within it. Pools act as containers for origin servers that store the hostnames or IP addresses of servers to which incoming requests can be directed. Each pool is attached to a version of a service, but origin servers are versionless and any changes will become effective immediately.

When Dynamic Servers might be useful

Dynamic Servers might be useful for organizations that need to load balance requests among origin servers. They can be used to:

- evaluate new server instance types and new software deployments.
- independently scale individual microservices.

More specifically, they can be used as:

- a **Local Server Load Balancer (LSLB)** where they are used to balance requests among origin servers within in a single region, such as AWS EC2 instances in the US East region, or within a single datacenter or on-premises location.
- a **Global Server Load Balancer (GSLB)** where they are used to load balance requests among origin servers across any geographically distributed infrastructure deployments such as:
  - within multiple regions of an Infrastructure as a Service (IaaS) provider (e.g., AWS, GCP, Microsoft Azure).
  - between multiple IaaS providers (e.g., AWS, GCP, Microsoft Azure).
  - as part of hybrid infrastructure deployments that include a combination of on-premises origin servers or datacenters and IaaS providers.

Getting started

You'll need to follow these steps:

1. Create a pool (/guides/dynamic-servers/creating-and-using-pools-with-dynamic-servers) in a working version of a service that's unlocked and not yet activated.
2. Add at least one origin server (/guides/dynamic-servers/creating-and-using-server-entries-with-dynamic-servers) to the newly created pool, keeping in mind the limitations.
3. Activate (/guides/basic-setup/working-with-services#editing-and-activating-versions-of-services) the version of the service you associated with the pool.

Once the pool is created, properly associated, and filled with origin servers, it can be called in your service.

Limitations

Keep the following limitations in mind as you use Dynamic Servers:

- **Each Fastly service can be configured with up to five origin servers.** Origin servers count as origins for the purposes of these limits. Contact sales@fastly.com (mailto:sales@fastly.com) to enable more than five origin servers per service in your account.
- **Pools cannot be created with custom VCL.** If you create a pool using the API, you can use the API to make changes to it and use custom VCL to interact with it.
- **Pools need at least one enabled server entry.** Origin servers cannot be deleted when a pool only has one enabled entry left.
- **Origin server deletions are permanent.** If you delete an origin server, it is permanently removed from all service versions and cannot be recovered.
- **When you delete a pool, you'll only delete it from the service version you're editing.** Pools are tied to versions and can be cloned and reverted. When using pools, we want you to be able to do things like delete a pool from a current version of your service in order to roll back your configuration to a previous version using as few steps as possible.
- **Event logs don't exist for origin server changes.** If you use the API to add, update, or remove an origin server, there will be no record of it. The only record of a change will exist when you compare service versions to view the point at which the pool was associated with the service version in the first place.

---

§ Creating and using pools with Dynamic Servers (/guides/dynamic-servers/creating-and-using-pools-with-dynamic-servers)

Fastly's Load Balancer allows you to create pools of origin servers (/guides/dynamic-servers/creating-and-using-server-entries-with-dynamic-servers) that you dynamically manage using Fastly's Dynamic Servers feature to distribute and direct incoming requests. A pool is responsible for balancing requests among a group of origin servers. In addition to load balancing, pools can be configured to attempt retrying failed requests. Pools have a quorum setting that can be used to determine when the pool as a whole is considered `up` in order to prevent problems following an outage as origin servers come back up.

> ❗ **IMPORTANT:** This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

Creating a pool

To start using Dynamic Servers, you'll need to create an empty pool within a version of a service that's unlocked and not yet activated. Make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/
version/<service_version>/pool -d 'name=<pool_name>&comment=<comment>'
```

The response will look like this:

```
{
    "between_bytes_timeout": "10000",
    "comment": "<comment>",
    "connect_timeout": "1000",
    "created_at": "2016-08-01T14:43:22+00:00",
    "deleted_at": null,
    "first_byte_timeout": "15000",
    "healthcheck": null,
    "id": "2IpWU5CGzPpbpGsABSDops",
    "max_conn_default": "200",
    "max_tls_version": null,
    "min_tls_version": null,
    "name": "<pool_name>",
    "quorum": "75",
    "request_condition": null,
    "service_id": "<service_id>",
    "shield": null,
    "tls_ca_cert": null,
    "tls_cert_hostname": null,
    "tls_check_cert": 1,
    "tls_ciphers": null,
    "tls_client_cert": null,
    "tls_client_key": null,
    "tls_sni_hostname": null,
    "type": "random",
    "updated_at": "2016-08-01T14:43:22+00:00",
    "use_tls": 0,
    "version": "<service_version>"
}
```


Be sure to activate (/guides/basic-setup/working-with-services#editing-and-activating-versions-of-services) the new version of the service you associated with the pool after adding at least one origin server (/guides/dynamic-servers/creating-and-using-server-entries-with-dynamic-servers).

ℹ **NOTE:** Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com (mailto:sales@fastly.com) to enable more than five origin servers per service in your account.

Viewing pools

To view a list of all pools attached to a particular version of a service, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/<service_version>/pool
```

The response will look like this:

```
[
{
    "between_bytes_timeout": "10000",
    "comment": "just my first pool",
    "connect_timeout": "1000",
    "created_at": "2016-08-01T14:43:22+00:00",
    "deleted_at": null,
    "first_byte_timeout": "15000",
    "healthcheck": null,
    "id": "2IpWU5CGzPpbpGsABSDops",
    "max_conn_default": "200",
    "max_tls_version": null,
    "min_tls_version": null,
    "name": "SP_Prod_Pool_1",
    "quorum": "75",
    "request_condition": null,
    "service_id": "<service_id>",
    "shield": null,
    "tls_ca_cert": null,
    "tls_cert_hostname": null,
    "tls_check_cert": 1,
    "tls_ciphers": null,
    "tls_client_cert": null,
    "tls_client_key": null,
    "tls_sni_hostname": null,
    "type": "random",
    "updated_at": "2016-08-01T14:43:22+00:00",
    "use_tls": 0,
    "version": "<service_version>"
}
]
```


To see information related to a single pool (in this example, `SP_Prod_Pool_1`) attached to a particular version of a service, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/
<service_version>/pool/<pool_name>
```

The response will look like this:

```
{
    "between_bytes_timeout": "10000",
    "comment": "just my first pool",
    "connect_timeout": "1000",
    "created_at": "2016-08-01T14:43:22+00:00",
    "deleted_at": null,
    "first_byte_timeout": "15000",
    "healthcheck": null,
    "id": "2IpWU5CGzPpbpGsABSDops",
    "max_conn_default": "200",
    "max_tls_version": null,
    "min_tls_version": null,
    "name": "SP_Prod_Pool_1",
    "quorum": "75",
    "request_condition": null,
    "service_id": "<service_id>",
    "shield": null,
    "tls_ca_cert": null,
    "tls_cert_hostname": null,
    "tls_check_cert": 1,
    "tls_ciphers": null,
    "tls_client_cert": null,
    "tls_client_key": null,
    "tls_sni_hostname": null,
    "type": "random",
    "updated_at": "2016-08-01T14:43:22+00:00",
    "use_tls": 0,
    "version": "<service_version>"
}
```


Deleting a pool

Deleting a pool deletes the pool and all of its associated server entries. To delete a pool (in this example, `SP_Prod_Pool_1`), make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id
>/version/<service_version>/pool/<pool_id>
```

The response will look like this:


```
{
  "status":"ok"
}
```

§ Creating and using server entries with Dynamic Servers (/guides/dynamic-servers/creating-and-using-server-entries-with-dynamic-servers)

Fastly's Load Balancer allows you to create pools (/guides/dynamic-servers/creating-and-using-pools-with-dynamic-servers) of origin servers that you dynamically manage using Fastly's Dynamic Servers feature to distribute and direct incoming requests. An origin server is an address (IP address or hostname) of a server to which the Dynamic Servers feature can forward requests. Fastly can then select any one of the origin servers based on a selection policy defined for the pool.

> ❗ **IMPORTANT:** This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.

Creating an origin server

To add an origin server to the pool, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/pool/<pool_id>/server -d 'address=<hostname_or_ip_address>'
```

The response will look like this:

```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```

ℹ **NOTE:** Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com (mailto:sales@fastly.com) to enable more than five origin servers per service in your account.

Viewing origin servers

To see information related to a single origin server, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/pool/<pool_id>/server/<hostname_or_ip_address>
```

The response will look like this:

```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```

To view a list of all origin servers attached to a particular pool, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/pool/<pool_id>/servers
```

The response will look like this:

```
[
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
]
```

Enabling and disabling origin servers

You can enable or disable an origin server to control whether or not traffic is sent to it. Disabling an origin server allows you to remove it from the pool temporarily.

Enabling an origin server

Origin servers are enabled by default. To enable an origin server that has been disabled, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/
pool/<pool_id>/server -d 'address=<hostname_or_ip_address>&disabled=false'
```

```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```

Disabling an origin server

To disable an origin server, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/
pool/<pool_id>/server -d 'address=<hostname_or_ip_address>&disabled=true'
```


```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": true,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```


Deleting an origin server

To permanently delete an origin server, make the following API call in a terminal application:

```
curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id
>/pool/<pool_id>/server/<server_id>
```

The response will look like this:


```
{
  "status":"ok"
}
```


NOTE: Pools must have at least one origin server. The API won't allow you to delete the last origin server in the pool.

- Guides (/guides/) > Load Balancer > Tutorials (/guides/load-balancer-tutorials/)

- Guides (/guides/) > Image optimization > Setup and use (/guides/imageopto-setup-use/)

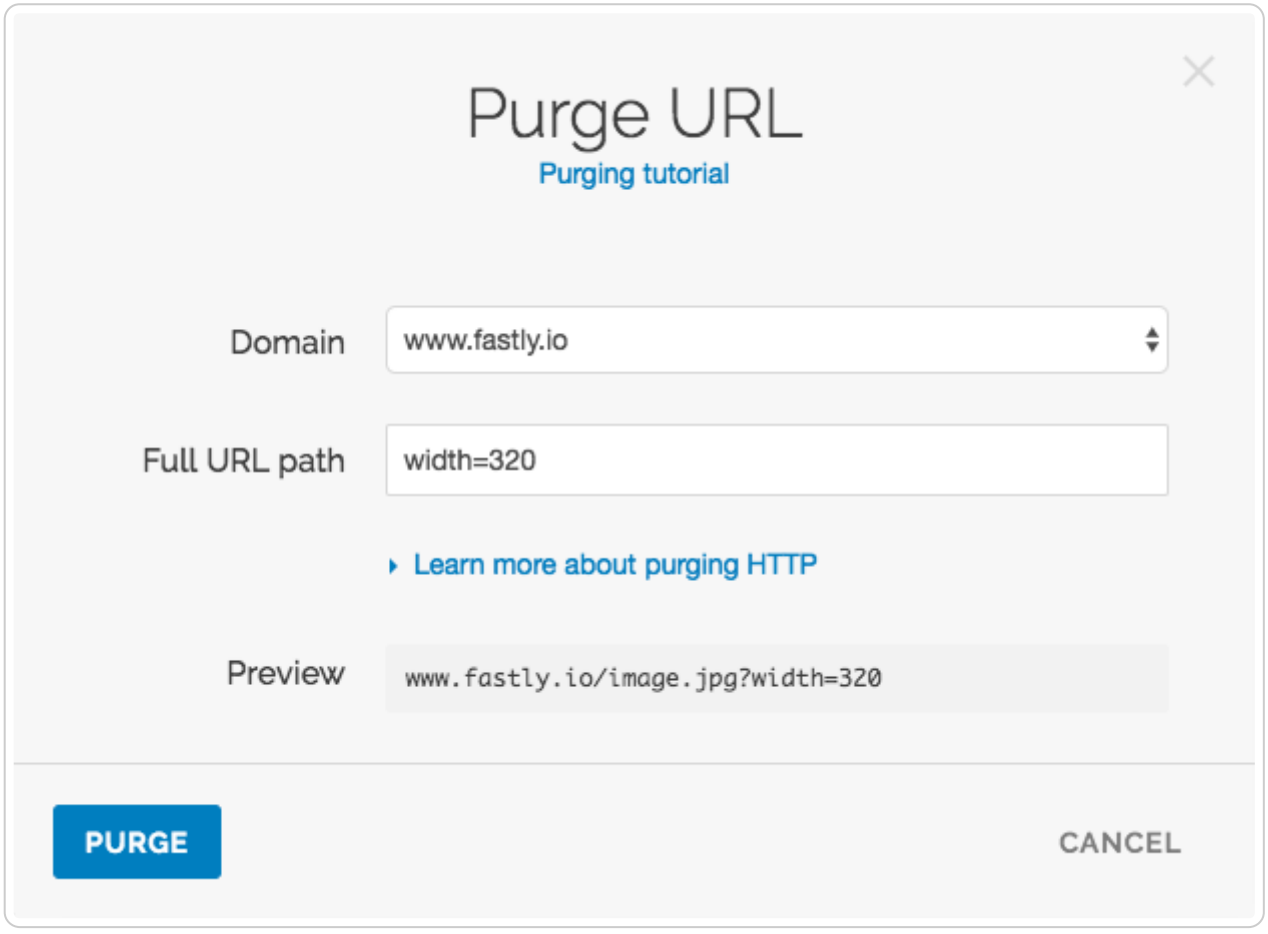
§ Purging optimized images (/guides/imageopto-setup-use/purging-optimized-images)

Instant Purging removes an image from Fastly caches immediately so it can be refreshed from your origin servers.

Purging images via the user interface

- Log in to the Fastly web interface and click the **Configure** link.
- From the service menu, select the appropriate service.
- From the **Purge** menu, select **Purge URL**. The Purge URL window appears.

Purge an individual image



- In the **Full URL path** field, type the path to the image you'll be purging (e.g., `/image.jpg?width=320`). The Preview field displays the URL that will be purged.
- Click the **Purge** button.

Purge all transformed image variations

- In the **Full URL path** field, type the path to the image removing all Fastly Image Optimizer API query string parameters. (e.g., `/image.jpg`). The Preview field displays the URL that will be purged.
- Click the **Purge** button.

Purging images via API

The syntax for purging a service through the API can be found in the Purging section (/api/purge) of the API (/api/) documentation.

Purge an individual image via API

To purge an individual image URL, type the path to the image you want to purge.

For example: `curl -X PURGE http://www.fastly.io/image.jpg?width=320`

Purge all transformed image variations via API

To purge all transformed image variations belonging to a specific image, remove all the Fastly Image Optimizer API query string parameters.

For example: `curl -X PURGE http://www.fastly.io/image.jpg`

> This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

---

§ Serving images (/guides/imageopto-setup-use/serving-images)

By adding the transformation URL API (/api/imageopto/#api) query string parameters to your existing image URLs, images can be resized, cropped, rotated, compressed, and transcoded into different image formats for increased performance. Image transformations can be applied programmatically and on-demand, eliminating the need to batch process or maintain multiple copies of an image to support different sizes and characteristics of device viewing your image content.

Example transformation

Resize an image to 200px wide.

`<img src="http://www.fastly.io/image.jpg?width=200"/>` (http://www.fastly.io/image.jpg?width=200)

Transformation order

Although the URL API (/api/imageopto/#api) parameters can be specified in any order, we normalize the transformation sequence within our system to the following order:

1. `trim`
2. `crop`
3. `orientation`
4. `width` `height` `dpr` `fit` `resize-filter` `disable`
5. `pad` `canvas` `bg-color`
6. `format` `quality`

Input and output formats

The source image can be any of the following image formats:

`GIF`, `PNG`, `JPEG`, `WEBP`

The optimized output image can be any of the following image formats:

`GIF`, `PNG-24`, `PNG-32`, `JPEG`, `WEBP`

Input and output limits

- The maximum input image file size is 50 Megabytes.
- The maximum input image dimensions are 12,000x12,000 pixels.
- The maximum output image dimensions are 8,192x8,192 pixels (8K Ultra HD).
- The maximum number of frames an animated GIF can contain is 1,000.

Default quality level

When no quality value is specified, a default value of 85 is automatically applied. The quality parameter will be ignored by GIF and PNG output formats.

Meta data removal

To optimize your images for delivery, all metadata (for example, EXIF, XMP, or ICC) is removed to reduce file size. If an image contains an ICC profile, the data is applied directly to the image to ensure color output is correct.

WebP image support

WebP images can be delivered to supported browsers by adding the `auto=webp` parameter.

> This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

---

§ Serving responsive images (/guides/imageopto-setup-use/serving-responsive-images)

The Fastly Image Optimizer allows you to generate optimized images for use in responsive websites. The examples below describe several common use cases to get you started implementing responsive images.

Adaptive device pixel ratios

Deliver a fixed-width image that can adapt to varying `device-pixel-ratios`.

```
<img srcset="http://www.fastly.io/image.jpg?width=320&dpr=1.5 1.5x,
             http://www.fastly.io/image.jpg?width=320&dpr=2 2x"
     src="http://www.fastly.io/image.jpg?width=320"/>
```

Learn about `srcset` browser support (http://caniuse.com/#feat=srcset) and specification (https://html.spec.whatwg.org/multipage/embedded-content.html#attr-img-srcset).

Art direction

Use the HTML5 `<picture>` tag to deliver different image crops at different browser viewport sizes.

```
<picture>
  <source srcset="http://www.fastly.io/image.jpg?width=600&crop=16:9" media="(min-width: 600p
x)"/>
  <img src="http://www.fastly.io/image.jpg?width=320&crop=1:1"/>
</picture>
```

Learn about `<picture>` browser support (http://caniuse.com/#search=picture) and specification (https://html.spec.whatwg.org/multipage/embedded-content.html#the-picture-element).

Type-switching

Use the best file format for the browser and allow graceful fallback for non-supporting formats.

```
<picture>
  <source type="image/webp" srcset="http://www.fastly.io/image.webp"/>
  <source type="image/png" srcset="http://www.fastly.io/image.png"/>
  <img src="http://www.fastly.io/image.jpg"/>
</picture>
```

Learn about `<picture>` browser support (http://caniuse.com/#search=picture) and specification (https://html.spec.whatwg.org/multipage/embedded-content.html#the-picture-element).

> This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

§ Setting up image optimization (/guides/imageopto-setup-use/setup)

> ⚠ **WARNING:** Only send image content through the Fastly Image Optimizer service. We bill for this traffic at a different rate than what we charge (/guides/account-types-and-billing/how-we-calculate-your-bill) for standard request traffic. Non-image content can't be optimized by this service and will cost you more.

To use the Fastly Image Optimizer complete these steps.

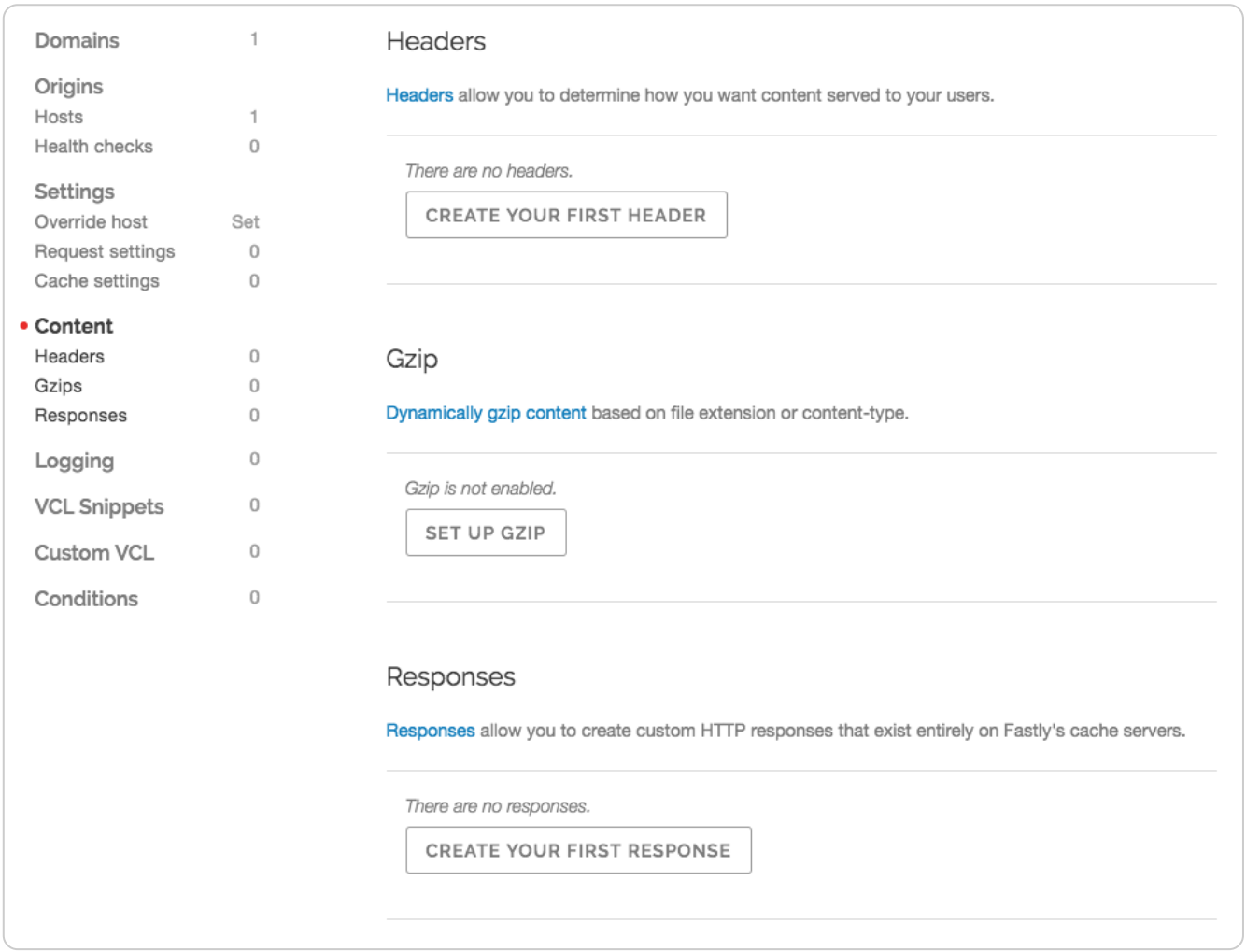
Contact Sales to request access

Contact sales (mailto:sales@fastly.com) to request access. Be sure to include the Service ID (/guides/account-management-and-security/finding-and-managing-your-account-info#finding-your-service-id) on which image optimization should be enabled.

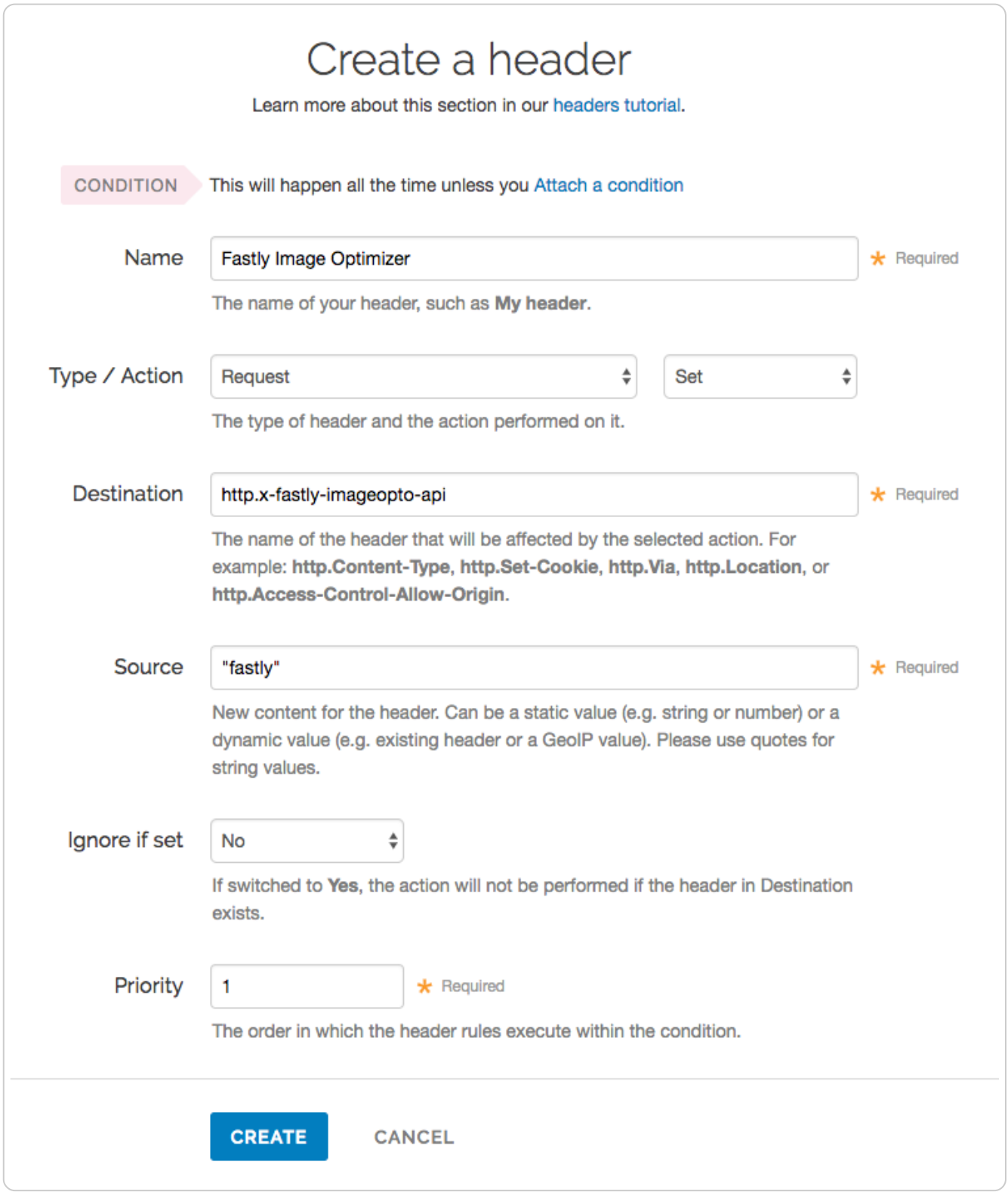
Add the Fastly Image Optimizer header

Once image optimization has been activated on your service ID and confirmed via email, configure your service by adding the Fastly Image Optimizer header.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.



5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** window as follows:
   - In the **Name** field, type `Fastly Image Optimizer`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.x-fastly-imageopto-api`.
   - In the **Source** field, type `"fastly"`.

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `1`.

7. Click **Create** to create the new header.

Passing additional query string parameters

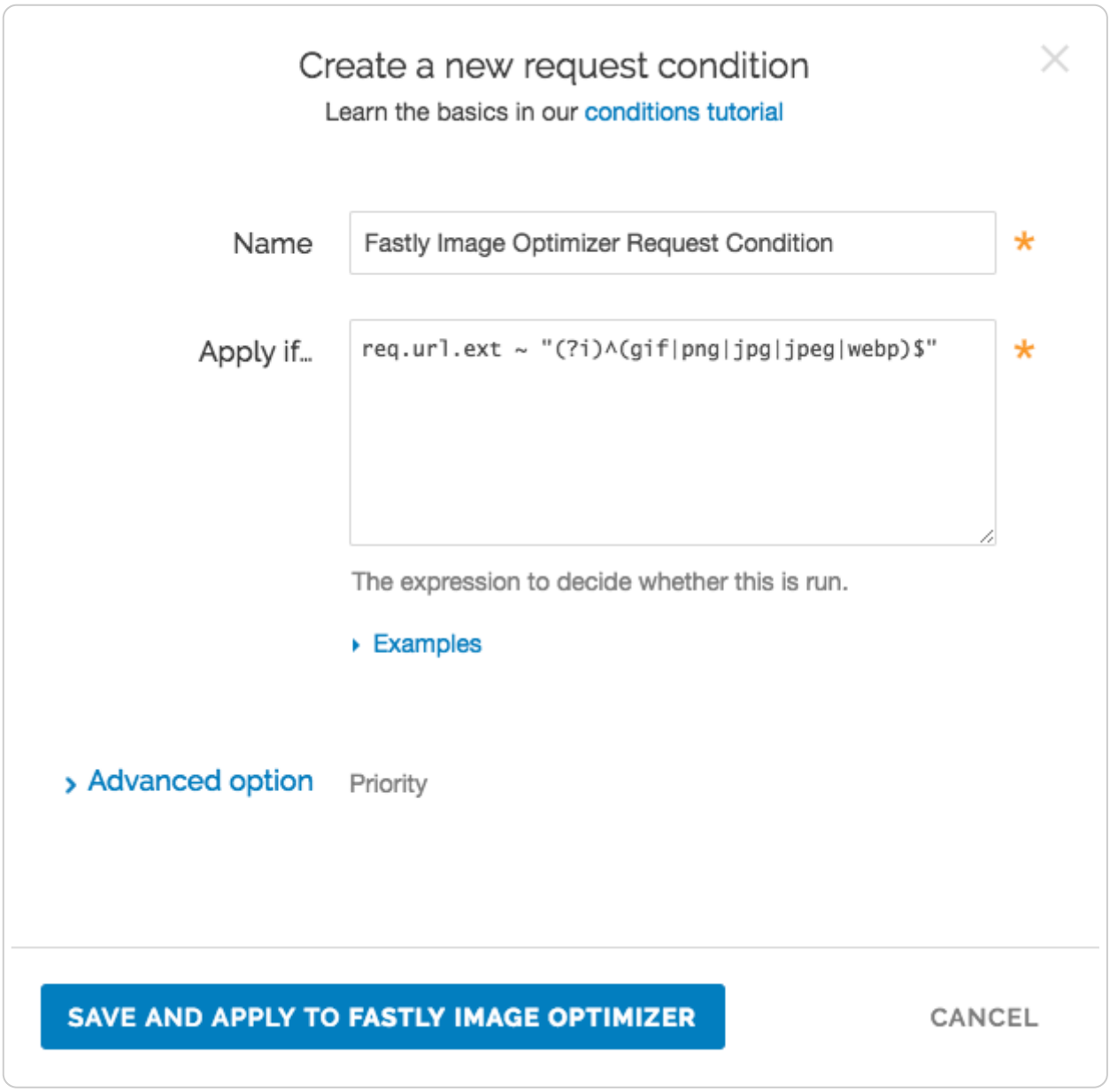
By default, the Fastly Image Optimizer removes any additional query string parameters that are not part of our image API (/api/imageopto/#image-api-reference). If, however your source image requires additional query string parameters to be delivered from origin then, in the **Source** field noted above, type `"fastly; qp=*"`.

> ★ **TIP:** For more help with adding or modifying headers, see our guide (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses).

Create a request condition

To ensure only your image assets are routed via the Fastly Image Optimizer, create a request condition.

1. Click the **Attach a condition** link next to the `Fastly Image Optimizer` header. The Add a condition window appears.
2. Click the **Create a new request condition** button. The Create a new request condition window appears.



3. Fill out the **Create a new request condition** window as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Fastly Image Optimizer Request Condition`).
   - In the **Apply if** field, type the appropriate request condition. For example, `req.url.ext ~ "(?i)^(gif|png|jpg|jpeg|webp)$"` will send all files with gif, png, jpg, jpeg, and webp extensions via the Fastly Image Optimizer. Likewise, `req.url ~ "^/images/"` will send all files in the `images` directory via the Fastly Image Optimizer.
4. Click the **Save and apply to** button to create the new condition for the header.

★ **TIP:** For more help using conditions, see our guide (/guides/conditions/using-conditions).

Enable shielding

To reduce cache miss latency and ensure long-lived connections, you must enable shielding for your origin. The shield location should be as geographically close to your image's origin as possible.

Our guide to enabling shielding (/guides/performance-tuning/shielding#enabling-shielding) provides more information on how to set this up. Take special note of the step immediately following your shielding location selection in that guide. If the Host header for the service has been changed from the default, you must ensure the new hostname is added to the list of domains.

Confirm everything is working

Once you've activated your changes, check to see if the Fastly Image Optimizer is processing your image request by typing the following cURL command on the command line:

```
curl -o image.jpg http://www.fastly.io/image.jpg?width=200 && convert image.jpg -print "Image Width: %wpx\n" /dev/null
```

Replace `http://www.fastly.io/image.jpg?width=200` with the full image URL and width of the image you're testing.

The command line output will display the image's width, which should match the width API parameter you added to your image. For example, the output might be:

```
Image Width: 200px
```

> This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

• Guides (/guides/) > Image optimization > Examples (/guides/imageopto-examples/)

§ Automating optimization (/guides/imageopto-examples/automating-optimization)

Encode to a JPEG and enable WebP automatic format selection to supported browsers

```
http://www.fastly.io/image.jpg?format=jpeg&auto=webp
```



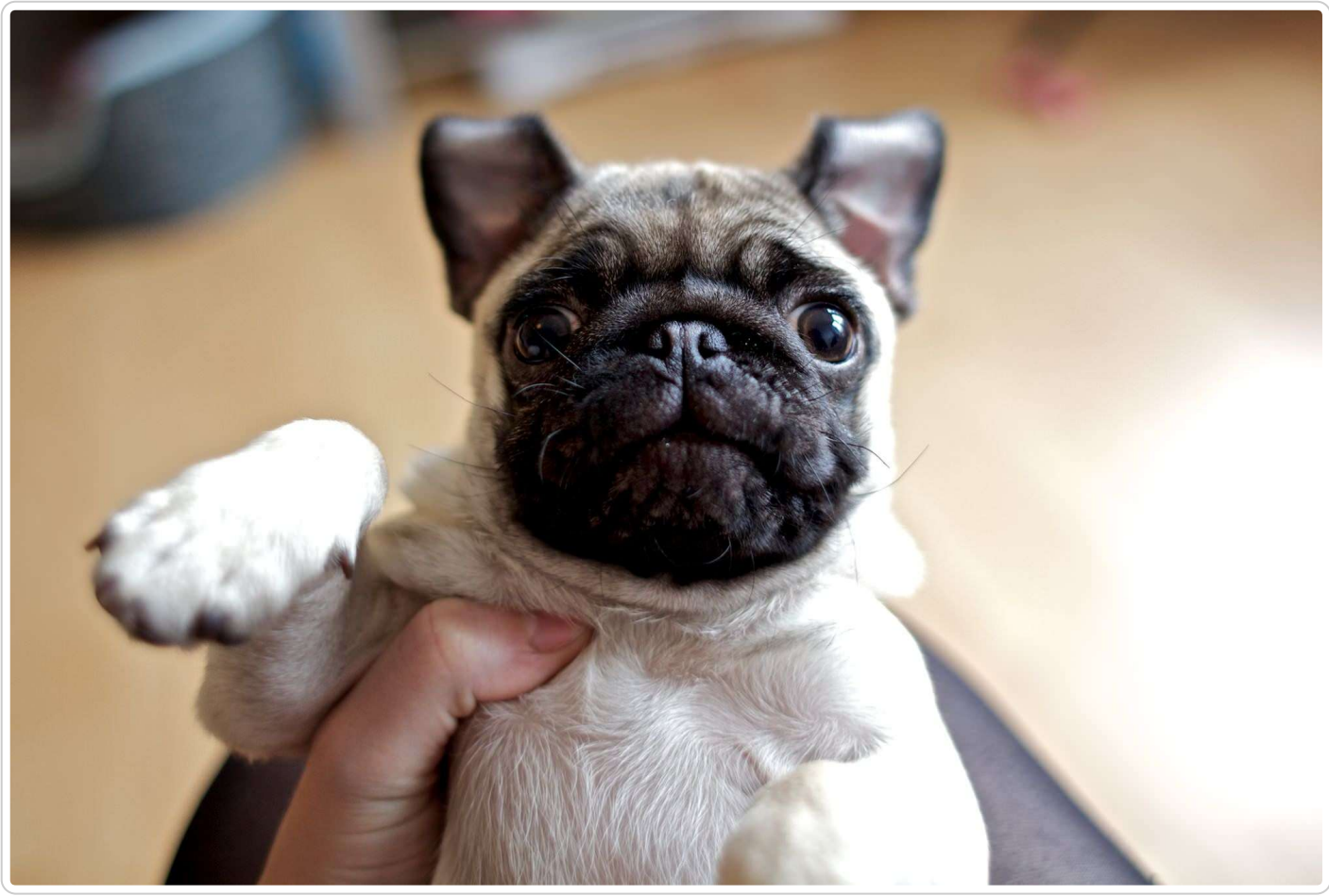
This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

§ Controlling image quality (/guides/imageopto-examples/controlling-image-quality)

Deliver an image at a specific % quality

Output an image with 60% compression (85 is the default for lossy images when IO is enabled).

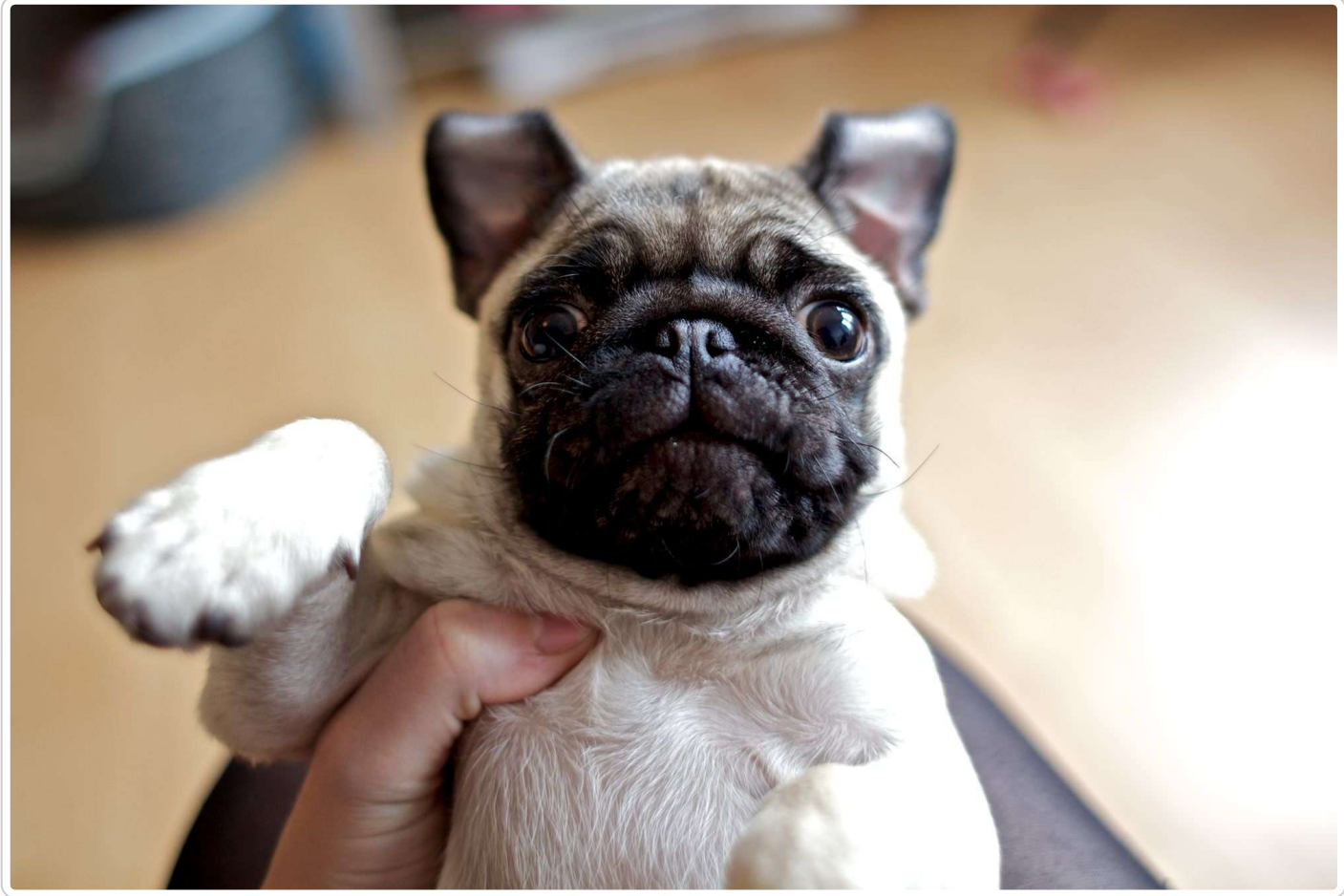
`http://www.fastly.io/image.jpg?quality=60`



Transcode and deliver an image at a specific % quality

Convert the image format to jpg and output and image with 60% compression.

`http://http://www.fastly.io/pug.png?format=jpg&quality=60`



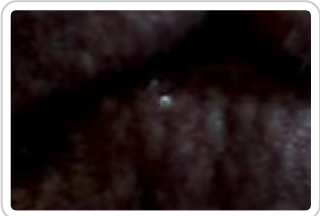
This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

§ Cropping images (/guides/imageopto-examples/cropping-images)

Region crop

Crop the image to 150px by 100px.

`http://www.fastly.io/image.jpg?crop=150,100`



Aspect ratio crop

Crop the image to an aspect ratio of 16:9.

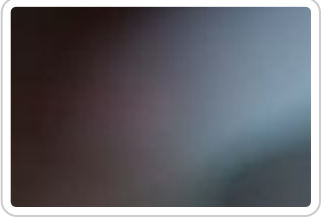
`http://www.fastly.io/image.jpg?crop=16:9`



Region crop and sub region

Crop the image to 150px by 100px and also select 50px as the starting sub region x coordinate and 50px as the sub region y coordinate.

`http://www.fastly.io/image.jpg?crop=150,100,x50,y50`



Aspect ratio crop and offset

Crop the image square and offset the x-axis 25% and the y-axis 50%.

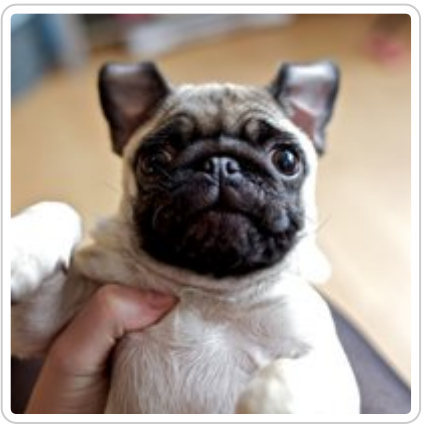
`http://www.fastly.io/image.jpg?crop=1:1,offset-x0.25,offset-y0.50`



Aspect ratio crop (with width)

Crop the image square and resize the width to 200px.

`http://www.fastly.io/image.jpg?crop=1:1&width=200`



This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

§ Reorienting images (/guides/imageopto-examples/reorienting-images)

Reorient the image right

`http://www.fastly.io/image.jpg?orient=r`





Flip the image horizontally

`http://www.fastly.io/image.jpg?orient=2`



Flip the image horizontally and vertically

`http://www.fastly.io/image.jpg?orient=3`



This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

§ Resizing images (/guides/imageopto-examples/resizing-images)

Pixel width resize

Resize the width to 200px.

```
http://www.fastly.io/image.jpg?width=200
```



Percentage height resize

Resize the height to 10% of the input image.

```
http://www.fastly.io/image.jpg?height=0.10
```



Percentage width resize over 100%

Resize the width to 150% of the input image.

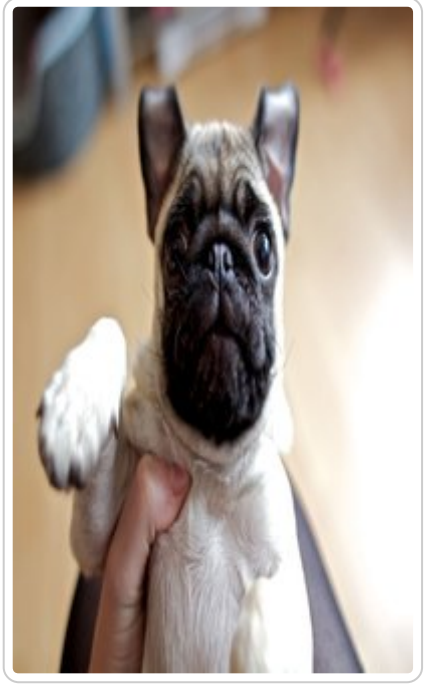
```
http://www.fastly.io/image.jpg?width=150p
```



Disproportionate resize

Disproportionally resize to a width of 200px and a height that is 25% of the original.

```
http://www.fastly.io/image.jpg?width=200&height=0.25
```



Resize to bounds

Resize the image to fit within the bounds of 150px in width by 150px in height.

```
http://www.fastly.io/image.jpg?width=150&height=150&fit=bounds
```



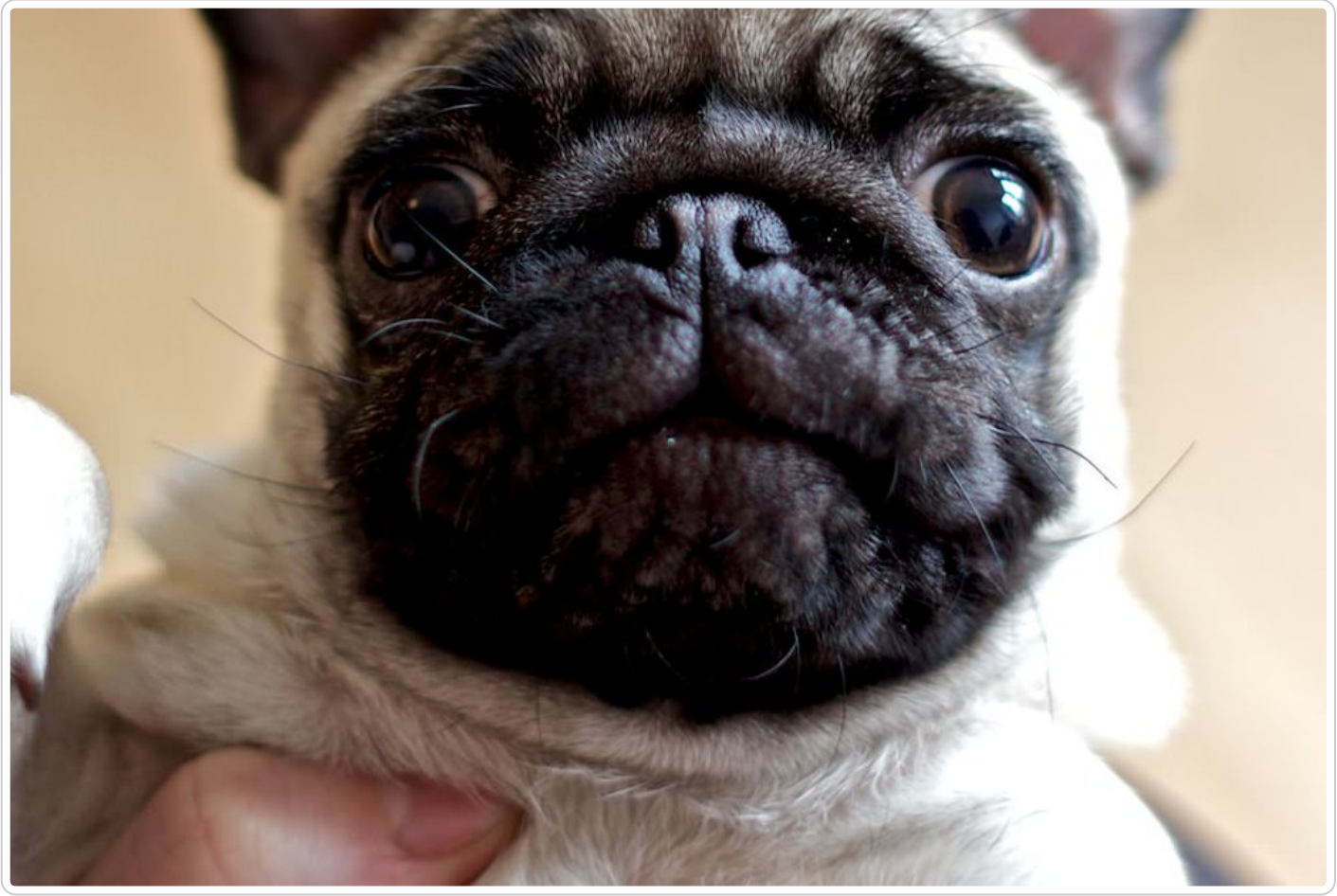
This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

§ Trimming images (/guides/imageopto-examples/trimming-images)

Trimming all edges by the same percentage

Trim all edges by 25%.

```
http://www.fastly.io/image.jpg?trim=0.25
```



Trimming parallel edges the same percentage

Trim top and bottom edge 25px, right and left edge 50px.

```
http://www.fastly.io/image.jpg?trim=25,50
```



Trimming all edges a different percentage

Trim top edge 25px, right edge 50px, bottom edge 75px and left edge 100px

```
http://www.fastly.io/image.jpg?trim=25,50,75,100
```



This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program (/guides/compliance/security-program#cloud-infrastructure-security-and-compliance-program).

§ Configuration guidelines for live streaming (/guides/live-streaming/configuration-guidelines-for-live-streaming)

The Fastly network can deliver live streams for any HTTP streaming technology (/guides/detailed-product-descriptions/about-fastlys-video-caching-service), archived or recorded, on any public or private cloud storage service. When configuring VCL (/guides/vcl/uploading-custom-vcl) to deliver

live streams, we recommend following these guidelines, which Customer Support (mailto:support@fastly.com) can help you with.

Configure shielding

Configure shielding (/guides/performance-tuning/shielding) by designating a specific shield POP for your origin to ensure live streams remain highly available within the Fastly network. If your setup includes primary and alternate origins (e.g., for high profile live streams), be sure to select a shield POP, close to each origin, one for each origin you define.

Configure video manifest and segment caching TTLs

In live streams, video manifests are periodically refreshed when new segments become available, specially for HLS. We recommend setting manifest file TTLs to less than half of the video segment duration, typically 1-2 seconds for 5-second video segments. For long DVRs and live-to-VOD transitions, set segment TTLs longer on shields and shorter on edge POPs such that they are served from memory (that is, less than 3600s).

The following VCL sample may help you implement differnet TTLs for video manifest and segments. It can also be added to your service using VCL Snippets (/guides/vcl-snippets/about-vcl-snippets):

```
sub vcl_fetch {
#FASTLY fetch

  # Set 1s ttls for video manifest and 3600s ttls for segments of HTTP Streaming formats.
  # Microsoft Smooth Streaming format manifest and segments do not have file extensions.
  # Look for the keywords "Manifest" and "QualityLevel" to identify manifest and segment reques
ts.
  if (req.url.ext ~ "m3u8|mpd" || req.url.path ~ "Manifest") {
    set beresp.ttl = 1s;
    return (deliver);
  }
  else {
    if (req.url.ext ~ "m4s|mp4|ts|aac" || req.url.path ~ "QualityLevel") {
      set beresp.ttl = 3600s;
      return (deliver);
    }
  }

  return (deliver);
}
```

Optionally, identify video manifests and segments using the MIME type.

Configure lower TTLs for errors

By default, Fastly honors the `Cache-Control` header from the origin to set TTLs for cacheable objects. However, origins may not send `Cache-Control` headers for responses with non-200 or 206 HTTP status codes. As a result, Fastly will cache these responses with default TTLs, usually 3600s, to prevent large numbers of requests from hitting an origin simultaneously. For live streams, new video segments are added every few seconds. Typically, transcoders are configured to generate 5s segments and manifests are refreshed after each new segment is available. Any requests to unavailable segments or errors on manifests would cache the response, with default TTLs resulting in errors to every request for those objects until the cache TTLs expire, even though the same objects could have become available much earlier. Setting lower cache TTLs for these error responses help origins recover more quickly.

The following VCL sample may help you implement this or can also be added to your service using VCL Snippets (/guides/vcl-snippets/about-vcl-snippets):

```
sub vcl_fetch {
#FASTLY fetch

  # Set 1s ttl if origin response HTTP status code is anything other than 200 and 206
  if (!http_status_matches(beresp.status, "200,206")) {
    set beresp.ttl = 1s;
    return (deliver);
  }

  return (deliver);
}
```

Configure Streaming Miss

Configure Streaming Miss (/guides/performance-tuning/improving-caching-performance-with-large-files#streaming-miss) to reduce the time clients (players) must wait to begin downloading streams when Fastly's edge servers must fetch content from your origin. Streaming Miss should be enabled for video or audio objects only (these are sometimes called "chunks" or "segments").

The following VCL sample may help you implement this. It can also be added to your service using VCL Snippets (/guides/vcl-snippets/about-vcl-snippets):

```
sub vcl_fetch {
#FASTLY fetch

  # Enable Streaming Miss only for video or audio objects.
  # Below conditions checks for video or audio file extensions commonly used in
  # HTTP Streaming formats.
  if (req.url.ext ~ "m4s|mp4|ts|aac") {
    set beresp.do_stream = true;
  }

  return (deliver);
}
```

Configure automatic gzipping

Configure automatic gzipping (/guides/basic-configuration/enabling-automatic-gzipping) for manifest files based on their file extension or content-type using the following table as a guide:

HTTP streaming format	**file extension**	**content-type**
Apple HLS	m3u8	application/x-mpegurl, application/vnd.apple.mpegurl
MPEG-DASH	mpd	application/dash+xml
Adobe HDS	f4m, bootstrap	application/f4m (for manifest), application/octet-stream (for bootstrap)
Microsoft HSS	N/A	application/vnd.ms-sstr+xml

Configure a CORS header

Configure a CORS (/guides/performance-tuning/enabling-cross-origin-resource-sharing) header on your service to play audio or video content on a different domain.

Enable TCP optimizations

Enable TCP tuning optimizations (e.g., CWND size) between cache servers and clients. For example, consider setting the CWND value higher, between 40-50. Enabling TCP optimizations allows clients to better estimate network bandwidth and thereby select the ideal rendition and bitrate (/guides/on-the-fly-packaging/adaptive-bitrate-playback-url-guidelines) for the best playback quality.

The following VCL sample may help you implement this. It can also be added to your service using VCL Snippets (/guides/vcl-snippets/about-vcl-snippets).:

```
sub vcl_deliver {
#FASTLY deliver

  # increase init cwnd
  if (client.requests == 1) {
    set client.socket.cwnd = 45;
  }

  return(deliver);
}
```

Configure origin timeouts

Set appropriate origin timeouts (/guides/debugging/changing-connection-timeouts-to-your-origin) to ensure new live video chunks and segments are fetched in a timely manner. Start by setting the First Byte (ms) and Between Bytes (ms) timeout values to less than half of the segment duration and then adjust the values accordingly to optimize them for your origin. For example, for a live stream configured with a 5s segment duration, you could start by setting both timeouts to 2000ms.

Consider setting up failover (fallback) origins

Consider configuring your VCL to allow your origins to failover (/guides/performance-tuning/failover-configuration) from high-profile primary streams to alternate streams in case of encoder failures or other issues (e.g., high resource utilization).

Configure real-time log streaming

For troubleshooting and debugging any end-to-end live streaming latency issues (also known as "hand-waving latency" or "glass latency"), configure real-time log streaming (/guides/streaming-logs/) and include TCP connection, caching, and different time-related metrics (/guides/vcl/date-and-time-related-vcl-features) in `vcl_log`. For example:

- `fastly_info.state` (cache hits or missess)
- `client.socket.tcpi_rtt` (client round trip time)
- `time.to_first_byte` (time from client request to the first byte being received)
- `time.elapsed` (time since the request started)
- `client.as.number` and `client.as.name` (autonomous system (https://en.wikipedia.org/wiki/Autonomous_system_(Internet)) number and name associated with client IP)
- `client.socket.tcpi_delta_retrans` and `client.socket.tcpi_total_retrans` (number of packets retransmitted to the client)

- `client.socket.nexthop` (network path Fastly is sending the client response)

These metrics can help you analyze throughput and may help you determine reasons a video player might switch quality levels during ABR playback (/guides/on-the-fly-packaging/adaptive-bitrate-playback-url-guidelines).

Take advantage of surrogate key purging

All video segments and the manifest for a live stream can be purged using a single API call by using Fastly's surrogate key feature (/guides/purging/getting-started-with-surrogate-keys).

Manage live-to-VOD smoothly

Most encoders generate a separate video manifest when making the same live stream available for VOD. If your VOD manifest has the same URL as the live one, purge the live stream video manifest or wait for the caches to invalidate (as they will be set with low TTLs). If your setup archives the live stream as progressive mp4s, consider delivering them using Fastly's OTFP service (/guides/detailed-product-descriptions/about-fastlys-onthefly-packaging-service).

> ℹ **NOTE: Wowza integrations.** When configuring your Wowza origin server, be sure to select the Live HTTP Origin (https://www.wowza.com/docs/how-to-configure-a-wowza-server-as-an-http-caching-origin) application type. If you select Live Edge, Wowza will always return a unique URL for manifest requests, resulting in extremely low cache hit.

---

§ Adaptive bitrate playback URL guidelines (/guides/on-the-fly-packaging/adaptive-bitrate-playback-url-guidelines)

Fastly's On-the-Fly Packaging (OTFP) service (/guides/detailed-product-descriptions/about-fastlys-onthefly-packaging-service) supports any directory structure you might use to store different quality levels of a video. To construct adaptive bitrate (ABR) playback URLs for a video, make directory paths to that video unique. Ensure all the files associated with a particular video (e.g., quality levels, subtitles) exist under a single directory.

For example, say you had a video called Example Video. Assuming you had multiple quality levels and associated files for Example Video, the following directory structure would provide the best start to constructing ABR playback URLs:

Directory path example	Description
`/foo/bar/example-video/`	Base folder unique to this video
`/foo/bar/example-video/480p_30fps.mp4`	Quality level 480p with 30 frames per sec with audio
`/foo/bar/example-video/720p_30fps.mp4`	Quality level 720p with audio with 30 frames per sec with audio
`/foo/bar/example-video/720p_60fps.mp4`	Quality level 720p with audio with 60 frames per sec with audio
`/foo/bar/example-video/1080p_30fps.mp4`	Quality level 1080p with audio with 30 frames per sec with audio
`/foo/bar/example-video/1080p_60fps.mp4`	Quality level 1080p with audio with 60 frames per sec with audio
`/foo/bar/example-video/4k_30fps.mp4`	Quality level 4k with audio with 30 frames per sec with audio

With this directory structure, the ABR playback URL for all videos in the base directory would follow this template:

```
http://example.com/path/to/dir/<video_id>/<quality_file1_name_wo_ext>,<quality_file2_name_wo_ext>,...,<quality_fileN_name_wo_ext>/master.<f4m|m3u8|mpd>
```

For example, the ABR playback URLs for Example Video in every format would be:

Format	Example URL
HDS	`http://example.com/foo/bar/example-video/480p_30fps,720p_30fps,720p_60fps,1080p_30fps,1080p_60fps,4k_30fps/master.f`
HLS	`http://example.com/foo/bar/example-video/480p_30fps,720p_30fps,720p_60fps,1080p_30fps,1080p_60fps,4k_30fps/master.m`
MPEG-DASH	`http://example.com/foo/bar/example-video/480p_30fps,720p_30fps,720p_60fps,1080p_30fps,1080p_60fps,4k_30fps/master.m`

You can reduce the duplication in ABR playback URLs separating out the repeated prefix and suffix info as follows:

`<filename_prefix><filename_variable><filename_suffix_wo_ext>.mp4`

and the template would change to one of the following:

```
http://example.com/path/to/dir/<video_id>/<filename_prefix><quality_file1_variable_name_wo_ext>,<quality_file2_variable_name_wo_ext>,...,<quality_fileN_variable_name_wo_ext>,<filename_suffix_wo_ext>/master.<f4m|m3u8|mpd>
```

```
http://example.com/path/to/dir/<video_id>/<filename_prefix><quality_file1_variable_name_wo_ext>,<quality_file2_variable_name_wo_ext>,...,<quality_fileN_variable_name_wo_ext>/master.<f4m|m3u8|mpd>
```

```
http://example.com/path/to/dir/<video_id>/<quality_file1_variable_name>,<quality_file2_variable_name>,...,<quality_fileN_variable_name>,<filename_suffix_wo_ext>/master.<f4m|m3u8|mpd>
```

IMPORTANT: To use token authentication (/guides/tutorials/enabling-token-authentication) with ABR manifest URLs, special modifications must be made using custom VCL (/guides/vcl/uploading-custom-vcl). Contact support@fastly.com (mailto:support@fastly.com) for assistance.

§ Collecting OTFP metrics (/guides/on-the-fly-packaging/collecting-oftp-metrics)

Fastly allows you to collect and process On-the-Fly Packaging (OTFP) service (/guides/detailed-product-descriptions/about-fastlys-onthefly-packaging-service) metrics for analysis using a combination of custom VCL updates and specific log streaming settings. Once you've set up OTFP metrics collection through remote log streaming you can use any of a number of third-party and open source software options to aggregate your logging data for visualization and further analysis.

Upload custom VCL

1. If you have not already done so, request the ability to upload custom VCL (/guides/vcl/uploading-custom-vcl) code be enabled for your account.
2. Review the caveats of mixing and matching Fastly VCL with custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl).
3. Add the following custom VCL to your Fastly VCL:

```
sub vcl_deliver {
  # Identify Request type
  if (req.url.ext ~ "m3u8|ts|aac|webvtt") {
    set resp.http.Otfp-Format = "HLS";
  } else if (req.url.ext ~ "mpd|m4s") {
    set resp.http.Otfp-Format = "DASH";
  } else {
    set resp.http.Otfp-Format = "OTHER";
  }

  # Extract name-value pairs Otfp Info herder
  if (resp.http.X-Fastly-Otfp-Info) {
    set resp.http.Otfp-SS = regsub(resp.http.X-Fastly-Otfp-Info, ".*ss=(\S+).*", "\1");
    set resp.http.Otfp-SL = regsub(resp.http.X-Fastly-Otfp-Info, ".*sl=(\S+).*", "\1");
    set resp.http.Otfp-VL = regsub(resp.http.X-Fastly-Otfp-Info, ".*vl=(\S+).*", "\1");

    # Resolution (rs name-value) not available for audio-only segments
    if (resp.http.X-Fastly-Otfp-Info ~ ".*rs=(\S+).*") {
      set resp.http.Otfp-RS = re.group.1;
    } else {
      set resp.http.Otfp-RS = "-";
    }
  }
#FASTLY deliver

  return(deliver);
}
```

Create a logging endpoint

Follow the instructions to set up remote log streaming (/guides/streaming-logs/setting-up-remote-log-streaming) for your account and when creating your specific logging endpoint, set the **Format String** field to the following:

```
%h now.sec %r %>s %b resp.http.Otfp-Format resp.http.Otfp-SS resp.http.Otfp-SL resp.http.Otfp-V
L resp.http.Otfp-RS
```

Control log file timing with a logging endpoint condition

To avoid excess log files, consider attaching a condition to the logging endpoint so logs are only sent when video segments are requested so that logging specifically exclude those files sent from Fastly's Origin Shield.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.

3. Click the **Configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Logging** link. The logging page appears.
5. In the list of logging endpoints, find the endpoint you enabled when setting up remote log streaming (/guides/streaming-logs/setting-up-remote-log-streaming), then click **Attach a condition**. The Create a new condition window appears.
6. Fill out the **Create a new condition** window as follows:
   - In the **Name** field, type a human-readable name for the condition.
   - In the **Apply if** field, type `resp.http.X-Fastly-Otfp-Info && !req.http.Fastly-FF`.
7. Click **Save and apply to**.

Analyze logging data

In addition to any Varnish variable (https://www.varnish-cache.org/docs/2.1/reference/vcl.html#variables), and a variety of Fastly's extensions to VCL (/guides/vcl/guide-to-vcl#fastlys-vcl-extensions), log files include the following video-specific fields:

- `ss` - video segment start presentation time in seconds
- `sl` - video segment duration in seconds
- `vl` - video duration in seconds
- `rs` - video track display resolution in pixels

You can use these fields to run queries for analysis and use what you discover to refine your video delivery settings.

• Guides (/guides/) > Compliance and law > About our terms (/guides/about-our-terms/)

• Guides (/guides/) > Compliance and law > Compliance (/guides/compliance/)

§ Security program (/guides/compliance/security-program)

Fastly's security program includes safeguards that help protect your data as it moves through the Fastly service. Information about these safeguards is organized by category. Our technology compliance (/guides/compliance/technology-compliance) guide describes additional safeguards we maintain.

Authentication and authorization

User account assignment. We assign individual user accounts to personnel who access Fastly systems and devices. These assignments help us monitor and enforce accountability of user activity.

User-level privileges. Our systems and devices enforce user roles or similar measures to control the extent of access we grant individual users.

Multi-factor authentication. We enforce multi-factor authentication to better secure our computing resources from unauthorized logins.

Application security

Secure software development. We provide annual training to Fastly developers to help identify and prevent common software vulnerabilities, including the OWASP Top 10 (https://www.owasp.org/index.php/OWASP_Top_Ten_Cheat_Sheet). Developer code undergoes peer review prior to deployment, and internal security engineers and third-party security validators periodically analyze code for software components with higher potential security risk.

Web application security review. A third party assesses the security of the Fastly web application annually. We address findings from this assessment according to the risk they pose to the security of the Fastly service.

Network and infrastructure security

Network security reviews. We regularly perform vulnerability scans and third-party penetration tests on the Fastly network. We review and address findings from these activities to help maintain the security of our network.

Configuration standards. We document and follow configuration standards to maintain secure systems and network devices. These standards include business justification for used ports, protocols, and services, as well as the removal of insecure default settings.

Vulnerability and patch management. To maintain awareness of potential security vulnerabilities, Fastly monitors public and private distribution lists, as well as reports submitted through our responsible disclosure (https://www.fastly.com/security/report-security-issue) process. We validate and implement security patches for critical vulnerabilities within 24 hours of discovery. For non-critical vulnerabilities and updates, we schedule and deploy vendor-provided patches on a regular basis.

Encryption

Secure data transmission. The Fastly service supports TLS configurations (/guides/securing-communications/) to encrypt connections both externally to end users and backend origin servers, as well as internally within the Fastly network.

Encryption key management. We maintain technology and procedures to secure private keys throughout their lifecycle.

Key storage and access security. We store private keys in encrypted repositories, and we restrict key storage access to personnel who support our key management processes.

Datacenter and physical security

Physical access restrictions. Our datacenters are fully enclosed with perimeter protection such as fences, gates, and mantraps to prevent unwanted entry. Only authorized people (including datacenter personnel, our employees, and contractors) may enter and move within a datacenter.

Datacenter access management. We ensure movement within our datacenters is monitored via onsite safeguards such as security guard assignment, facility access logging and review, and video surveillance. Additionally, we periodically review and adjust the list of personnel who may enter our datacenters.

Secure asset installation. We install computer and network hardware in locked cages and racks. Only authorized individuals may physically access this equipment.

Environmental safeguards. Our datacenters compensate for environmental disruptions with systems that control backup power, temperature and humidity, and fire suppression.

Business continuity and operational resilience

Service failover. If any of our points of presence (POPs (/guides/basic-concepts/fastly-pop-locations)) experience issues serving content, we can redirect traffic to a neighboring POP without interrupting the delivery of content to end users.

Internet redundancy. Our datacenters have connections with multiple Internet service providers. We do not rely on any single carrier to serve content to end users.

Service monitoring. We monitor multiple internal and external reporting channels to detect service-related issues. Personnel are available 24x7x365 to confirm and respond to disruptions of the Fastly service.

Communication and reporting. We update impacted customers using various communication methods (such as status.fastly.com (https://status.fastly.com/)), depending on an incident's scope and severity.

Security incident management

Incident response plan. We maintain a formal incident response plan with established roles and responsibilities, communication protocols, and response procedures. We review and update this plan periodically to adapt it to evolving threats and risks to the Fastly service.

Incident response team. Representatives from key departments help address security-related incidents we discover. These personnel coordinate the investigation and resolution of incidents, as well as communication with external contacts as needed.

Breach notification. Fastly will notify affected customers within 48 hours of validating an unauthorized disclosure of customer confidential information.

Logging and monitoring

Log analysis. We aggregate and securely store Fastly internal system activity. Monitoring these logs helps us discover and investigate potential security issues.

Change and configuration monitoring. We use multiple monitoring and alert mechanisms to enhance the visibility of technology changes and help ensure adherence to our change management process.

Intrusion detection. We maintain mechanisms to detect potential intrusions at the network and host level. Our Security department inspects and responds to events these detection measures discover.

Customer and end user data management

Cache data and configurations. Customers manage which content is cached, where, and for how long by setting policies that control that content. See our introduction to caching (/guides/basic-concepts/how-fastlys-cdn-service-works) for more information. We may directly access or modify customer accounts or configurations to provide our services, prevent or address service or technical issues, as required by law, or as customers expressly permit. For the same reasons, we may also access or modify equipment, systems, or services that manage customer content.

Client IP addresses. As part of our caching network's general interaction with the Internet, Fastly independently collects anonymized and aggregated client IP address information on a limited basis to provide and improve its services. Client IP addresses are retained in a non-anonymized, non-aggregated fashion for up to two business days, or up to seven days if those addresses are associated with transmission errors (such as 503 "Service Unavailable" errors (/guides/debugging/common-503-errors)), and are discarded thereafter.

Subscriber IP addresses. Fastly independently collects the IP addresses of users who access their services within the Fastly web interface or through the API. We make these IP addresses available to customers through our event log functionality (/guides/monitoring-and-testing/monitoring-account-activity-with-event-logs). If customers define origin servers or syslog endpoints with IP addresses, we save those IP addresses as part of their configurations. We may

retain IP addresses from event logs or configurations indefinitely. Dynamically-resolved origin IP addresses may be retained for up to two business days, or up to seven days if those addresses are associated with transmission errors (such as 503 "Service Unavailable" errors (/guides/debugging/common-503-errors)), and are discarded thereafter.

IP addresses and security monitoring. Fastly may retain indefinitely any non-anonymized, non-aggregated client or subscriber IP addresses associated with suspicious activity that may pose a risk to the Fastly network or our customers, or that are associated with administrative connections to the Fastly service.

Content request data. Content enters, transits, and departs our network in response to requests. We retain and use data about the operation and reliability of our processing of requests to monitor, maintain, and improve our services, our business operations, and our security and compliance programs. Subject to confidentiality obligations to our customers, we only disclose this data in anonymized and aggregated form.

Subscriber log streaming. Subscribers may stream syslog activity (/guides/streaming-logs/), including end user IP addresses (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses#common-sources-of-new-content), to a remote endpoint for analysis and use. Fastly does not retain subscriber syslog activity, except as described above.

Cloud infrastructure security and compliance program

The use of third-party cloud infrastructure to host Fastly products that deliver content or process requests requires us to address certain aspects of our security and technology compliance programs differently from when Fastly directly manages the infrastructure.

Datacenter and physical security. For cloud infrastructure we use, Fastly relies on datacenter space under the control of the cloud infrastructure providers. These providers may have physical access to assets that contain data from Fastly services. As part of our third-party security review (/guides/compliance/technology-compliance#governance) process, we confirm that these providers maintain appropriate physical security measures to protect their datacenter facilities.

Business continuity and operational resilience. We deploy cloud-hosted products in multiple infrastructure regions or zones to help maintain those services when operational issues occur. If failure of a service occurs within a single availability zone, Fastly will automatically attempt to use cloud nodes in another zone.

Encryption. Fastly leverages in-transit and at-rest encryption to help secure data sent between Fastly and the cloud infrastructure provider or to secure data that resides on cloud infrastructure. Because we use at-rest encryption features offered by infrastructure providers, those providers may also hold the private encryption keys. As part of our third-party security review process, we confirm that these providers maintain secure encryption key management processes.

§ Technology compliance (/guides/compliance/technology-compliance)

Fastly's technology compliance program includes safeguards that help protect your data as it moves through the Fastly service. Information about these safeguards is organized by category. Our security program (/guides/compliance/security-program) guide describes additional safeguards we maintain.

Governance

Information security roles and responsibilities. We have formally assigned information security duties to Fastly personnel. Our Chief Security Officer and Security organization work with other departments to safeguard sensitive information related to the Fastly service.

Policies and procedures. Our policies and procedures help us maintain security in our systems, processes, and employee practices. Fastly's Security organization formally reviews these policies and procedures at least annually.

Risk management. We integrate risk assessment activities with various processes to identify and address information security risk to the company and customer data on our network.

Vendor security oversight. Fastly performs risk-based evaluations of the security measures of our vendors. We review these security measures before we begin using a vendor, and we ask the vendor to formally acknowledge these measures. We re-evaluate vendor security measures on a recurring basis thereafter.

Human resources security

Employee background screening. We screen new employees as part of the hiring process. Screening activities depend on applicable local regulations and may include criminal background checks and reference checks.

Confidentiality agreement. Our employees formally agree to safeguard the sensitive information they may view, process, or transmit as part of their job functions.

Security awareness training. We train our people to protect the data and devices they use. Each employee receives security awareness training as part of new hire procedures, and current employees take this training annually.

Data privacy

Privacy policy. Our privacy policy (https://www.fastly.com/privacy) describes how we collect, use, and protect the personal information of customer personnel using our websites or configuring services in the Fastly web interface. We certify our privacy policy and practices with TRUSTe (https://privacy.truste.com/privacy-seal/Fastly/validation?rid=e864daae-e54e-4aba-9ae0-6a76f78e0962).

Personal data transfer. The Fastly services by default do not process personal data. However, our service can be configured or used at the direction of the customer to process personal data. Our guide about our terms (/guides/about-our-terms/#can-i-transfer-personal-data-from-the-eu-to-the-us-using-the-fastly-services) provides additional information about data privacy compliance related to the processing of personal data.

Technology change management

Change management process. We follow a defined set of procedures to develop and deploy technology changes. These changes include updates to software, configurations, and devices that support the Fastly service.

Testing. We test technology changes at various stages of development, and we confirm those tests are successful before completing a deployment into the Fastly service.

Change approval and notification. As part of our deployment process, we prepare, approve, and communicate change notices to maintain awareness among personnel who manage the Fastly network and systems.

Post-implementation review. We confirm the success of changes after their deployment. Should we experience issues during implementation, we also maintain procedures to revert changes.

Identity and access management

User requests and approval. We document and approve requests for user access to the Fastly network. Our security administrators confirm appropriate documentation is in place before granting requested user rights.

Access modification. We promptly update or remove an employee's access to the Fastly network to match that employee's current job function or employment status.

User access review. We periodically inspect access privileges to make sure our personnel have appropriate access to Fastly systems and data.

• Guides (/guides/) > Compliance and law > Security measures (/guides/security-measures/)

• Guides (/guides/) > Compliance and law > Third-party technology (/guides/third-party-technology/)

• Guides (/guides/) > Compliance and law > Cloud-hosted products (/guides/cloud-hosted-products/)

• Guides (/guides/) > Compliance and law > Related offerings (/guides/related-offerings/)

• Guides (/guides/) > Compliance and law > Translations (/guides/translations/)

• Guides (/guides/) > Compliance and law > Archives (/guides/archives/)

• Guides (/guides/) > Products and services > Detailed product descriptions (/guides/detailed-product-descriptions/)

§ About Fastly's Application Programming Interface (API) (/guides/detailed-product-descriptions/about-fastlys-application-programming-interface)

Fastly provides an application programming interface (API) that can be accessed via a number of popular interactive clients. The Fastly API allows you to manage Fastly services via remote procedure calls instead of the web interface (/guides/basic-concepts/about-the-web-interface-controls). This currently includes features such as:

- Authentication (/api/auth)
- Configuration (/api/config)
- Historical Stats (/api/stats)

- Purging (/api/purge)
- Remote Logging (/api/logging)

The API features do not include customer account setup, which can only occur through the web interface controls. For examples of each API call in action, including full descriptions of the fields used and examples of requests and responses, see Fastly's API Reference (/api/).

Available API clients

The API's main entry point is https://api.fastly.com (https://api.fastly.com). It can be accessed via the following interactive clients:

- a Perl module
- a Ruby gem
- two different Python libraries
- a Node.js client
- a Scala client

Fastly's API Client web page (/api/clients) contains links to GitHub repositories where these clients can be found. When third-party organizations have supplied these clients, we've noted so on the web page. We also have several integrations (/guides/integrations/) available.

> Fastly makes no warranty on third-party software. We assume no responsibility for errors or omissions in the third-party software or documentation available. Using such software is done entirely at your own discretion and risk.

Authentication via the API

Nearly all API calls require requests to be authenticated. API tokens (/guides/account-management-and-security/using-api-tokens) allow you to create unique authentication identifiers for the users and applications authorized to interact with your service. You can scope the authorization of API tokens to a single service, and you can restrict access to purge all (/guides/purging/single-purges#purging-all-content). See the API authentication web page (/api/auth) for additional information.

---

§ About Fastly's Cloud Accelerator (/guides/detailed-product-

descriptions/about-fastlys-cloud-accelerator)

Fastly's Cloud Accelerator provides an integration between Fastly and Google services. Specifically, the integration enables you to connect to Google's Cloud Platform (https://cloud.google.com) service via peered network interconnections (direct PNIs) directly to Fastly's content delivery network services, thus speeding up your content delivery and optimizing backend workload.

When you sign up for Fastly services (/guides/basic-setup/sign-up-and-create-your-first-service) and configure a Google Cloud Platform service as your origin server, you designate a specific point of presence (POP) to serve as an Origin Shield that handles cached content (/guides/basic-concepts/how-fastlys-cdn-service-works) from their servers.

Requests from Fastly POPs (/guides/basic-concepts/fastly-pop-locations) to the Cloud Accelerator Origin Shields are routed over Fastly's network, which leverages optimized TCP connection handling, quick-start and opened connections to ensure fast response times between POPs and through to the end-user. Fastly ensures that requests go directly to the Origin Shield instead of the origin servers. Only requests that the entire network has never handled will go back to the Google Cloud Platform service.

§ About Fastly's Customer Support (/guides/detailed-product-descriptions/about-fastlys-customer-support)

Fastly offers standard technical support for all accounts through its support@fastly.com (mailto:support@fastly.com) email address. However, support availability and response times vary depending on the type of account you have and the level of support you have purchased. Our support description and SLA (/guides/detailed-product-descriptions/support-description-and-sla) information discusses these details further.

For all levels of technical support, Fastly's customer support ticketing system automatically generates and assigns each request a unique service ticket number, which is then sent within minutes to the customer at the email address used when submitting the ticket. This automated

response also contains a direct link to the service ticket within Fastly's customer support system. Fastly then responds to these requests as appropriate for the level of support purchased by the customer.

Accounts with Gold or Platinum Support

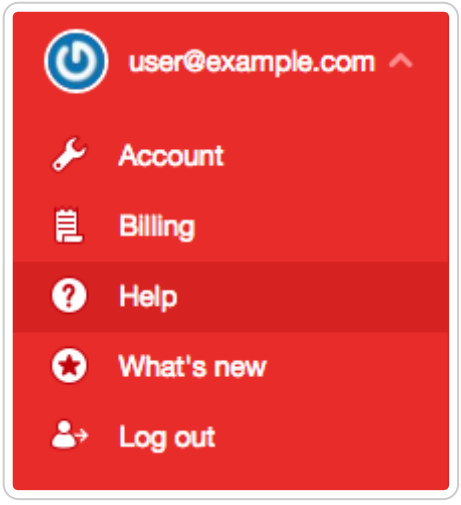
To submit emergency support tickets, customers must pay for a Gold or Platinum Support plan that includes support for severe incidents. Customers with Gold or Platinum Support plans can submit requests for support any time:

- a service that has been working stops working.
- a production deployment experiences a drastic change in performance that either is customer- or end-user-affecting.

Customers with Gold or Platinum Support accounts submit emergency tickets to an emergency support email address (instead of via the standard Customer Support email address (mailto:support@fastly.com)). Customers with Platinum Support accounts can also submit tickets via a toll-free telephone number.

Accounts without Gold or Platinum Support

Customers who do not pay for Gold or Platinum Support (/guides/detailed-product-descriptions/support-description-and-sla) levels can submit support requests via email to support@fastly.com (mailto:support@fastly.com) or directly in the Fastly web interface, via the Help link in the user menu, any time they require assistance with self-provisioned tasks (/guides/basic-concepts/self-provisioned-fastly-services). Fastly then responds to these requests as appropriate for the level of support purchased by the customer.



Tips on what to include in a support request

To help shorten the time it takes to resolve a submitted support request, here are some tips on key information to include in your request when you reach out to the Fastly support team:

- **Give us your customer ID if you have one.** Your customer ID (/guides/account-management-and-security/finding-and-managing-your-account-info#finding-your-customer-id), service ID, or the email address associated with your Fastly account help us identify who you are. If those don't apply, any information that would help us verify your identity will help us get started on your issue more quickly.
- **Tell us what's happening.** This helps us better understand your issue. For example, you expected to see something in your service and it's not there, or you're seeing an error message. Be sure to include the same information entered on the subject line within the body of the support request.
- **Describe your problem in detail.** We need information like specific service IDs and relevant URLs that aren't behaving as expected, the exact behavior you observed (perhaps include screenshots of what you saw), and whether or not you recently made changes that might have contributed to the issues (e.g., new DNS entries or new origins). These details help us focus support where it's needed most and makes our response to you faster.
- **Provide the steps you took.** This allows us to quickly reproduce your issue. For example, send the cURL command you ran if it created an issue or if you were trying a new configuration change, indicate whether you were using the API, the web interface, or custom VCL. Also, be sure to remove any passwords and API tokens from the content you provide in the support request.
- **Use Fastly Debug to identify network issues.** Supplying the output of Fastly Debug (http://www.fastly-debug.com/) provides us with key diagnostic information. For example, your requests are being routed to a POP you didn't expect, the debugging output helps us understand what's happening between the browser and Fastly POPs.

§ About Fastly's Full-site Delivery features (/guides/detailed-product-descriptions/about-fastlys-full-site-delivery-features)

Fastly offers full-site delivery features that allow you to speed up websites and mobile apps by pushing content closer to users, providing improved and secure experiences across the world.

HTTP request fulfillment

The Fastly CDN Service responds to HTTP GET requests (/guides/basic-concepts/content-and-its-delivery) initiated from end users' using your website, or from a program making calls to an internet-hosted API.

Header support

Fastly's CDN Service supports forwarding HTTP headers (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) to end users when they are received from your origin server. Alternatively, headers can be added, removed, or modified using our edge scripting language either before or after caching a response from the origin. This includes the Cache-Control and Expires headers as well as the Surrogate-Control header. HTTP header support allows you to send one set of instructions to the Fastly cache servers and another set of instructions to downstream caches, such as proxies or browsers. In particular, the Surrogate-Control header allows you to specify how to forward and transform specific header types.

Time to Live support

Fastly has no set hard limit on how long objects will remain cached (/guides/performance-tuning/controlling-caching#how-long-fastly-caches-content). Instead, Fastly supports the expiration of content via Time to Live (TTL) settings that you configure. TTL settings work as timers on your cached content. When content has resided in the cache for the entire TTL interval, that content is given the status of "expired." Before Fastly delivers requested content that is expired, the cache checks to see if the content is still valid by checking with your application server first.

If the application server says the content remains unchanged, the cache sets the content's status to "valid" and resets its TTL value. If the object has been changed, it is declared "invalid" because the content has expired. The application server delivers updated content. Fastly's CDN Service caches the updated content with the status of "valid," and its TTL timer begins to run.

The fetch and refresh process may take a second or more, and during that time, a Fastly cache may receive dozens or hundreds of end-user requests for that content. Fastly's request collapsing feature groups these requests and fulfills them at once when the application server response is received.

Fastly offers you the option of setting a global, default TTL for cached content control. When set, Fastly's CDN service caches objects in a consistent manner even when applications are inconsistent in doing so.

Origin shielding

When configuring Fastly's CDN Service during the self-provisioning process (/guides/basic-concepts/self-provisioned-fastly-services), you can designate a specific point of presence (POP) to serve as a shield for your origin servers. This server is referred to as a "shield" because it

protects your application servers from continuous requests for content. By default, no origin shield is enabled for you. You must specifically enable shielding (/guides/performance-tuning/shielding) to use it.

If Fastly's caches do not have the content being requested, they fetch it from the shield server instead of your origin servers. Fastly caches fetch content from your origin server only when the shield server does not have the content being requested.

Load balancing

You can designate multiple servers as your origin servers. When two or more application servers are provisioned as origin servers, Fastly's CDN Service will distribute requests to fetch content across those application servers using the round-robin method of distribution. This type of load balancing (/guides/performance-tuning/load-balancing-configuration) is enabled by default. You must explicitly disable it if you don't want to use it.

Request collapsing

Cached content sometimes must be refreshed when that content becomes "stale" or expires. When multiple end users request content that is in the process of being refreshed, request collapsing (/guides/performance-tuning/request-collapsing) groups those requests to be satisfied together, as soon as the content is received. This accelerates content delivery by keeping Fastly's CDN Service from repeating duplicate requests to your origin server. Request collapsing is enabled by default.

Instant Purge support

Fastly supports an Instant Purge feature that allows you to actively invalidate content (/guides/purging/). Rather than requiring your network operations and application staff to guess how frequently each bit of content may change, Fastly allows you to generate an HTTP Purge method that is sent to the CDN Service whenever an application changes or deletes data in its database. The Fastly CDN Service invalidates the associated content throughout the service's cache network, causing a new version of that content to be retrieved from the application server the next time it is requested.

Fastly allows URL-based and key-based purging, as well as purging of all content at once via specific, configurable purging commands (/api/purge). Fastly currently supports Ruby, Python, PHP, and Perl libraries (/api/clients) for instant purging.

When purging by URL or surrogate key, Fastly's CDN Service can process thousands of changes per second. The invalidation process takes less than 300 milliseconds, making it possible to deliver dynamic content that changes rapidly and unpredictably. Using Instant Purge, you can eliminate cache-to-origin HTTP traffic that all other CDN services generate to determine if expired objects are still valid.

Health checks

You have the option to configure Fastly's CDN Service to perform health checks (/guides/basic-configuration/health-checks-tutorial) on your application servers and measure their responsiveness. You can use health check responsiveness measurements to fine-tune the distribution of fetch requests. Health checks are not enabled by default. You must specifically enable them.

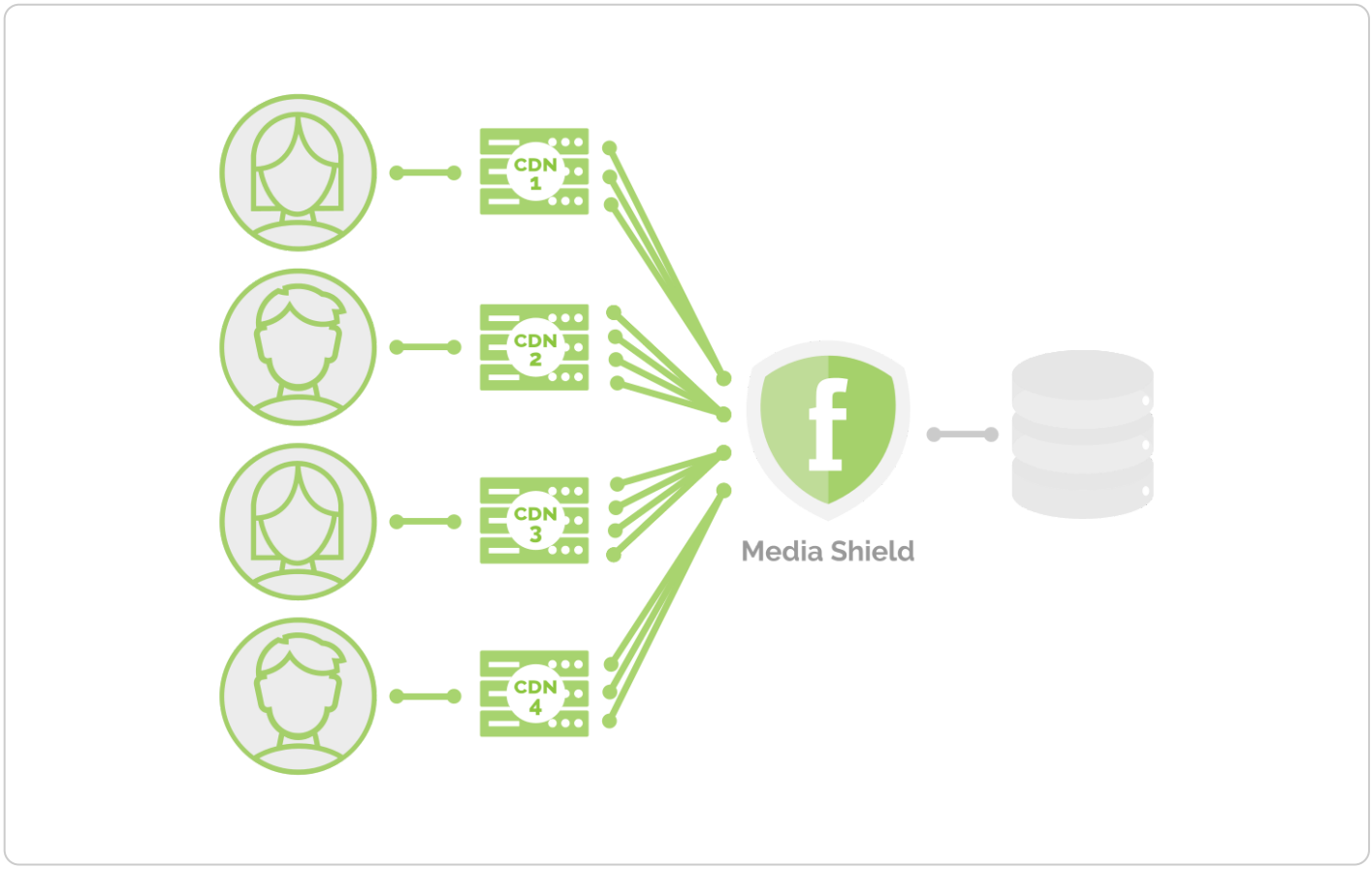
Grace mode support

When an application server becomes unavailable for any reason, end users will normally receive error messages indicating the content they've requested cannot be retrieved. When enabled, grace mode shields application servers by instructing Fastly's CDN Service to continue to serve stale or expired (but likely still valid) content to end users for a set amount of time. This allows you to return otherwise unavailable application servers to normal operations and still serve content rather than error messages to end users. By default, grace mode is not configured. You must specifically configure you service to serve stale content (/guides/performance-tuning/serving-stale-content) to use grace mode.

---

§ About Fastly's Media Shield (/guides/detailed-product-descriptions/about-fastlys-media-shield)

❗ **IMPORTANT:** This feature is part of a limited availability release. For more information, see our product and feature lifecycle (/guides/fastly-product-lifecycle/#limited-availability) descriptions.



Fastly offers large video streaming customers the ability to decrease origin traffic by reducing multiple CDN requests down to a single request back to origin. Media Shield is designed to work with your existing workflows by making Fastly the origin to all of your end-user-serving CDNs. Specifically, Media Shield combines Fastly's origin shielding (/guides/performance-tuning/shielding) and request collapsing (/guides/performance-tuning/request-collapsing) functionality, along with DNS mapping, to direct traffic to a subset of Fastly POP locations (/guides/basic-concepts/fastly-pop-locations).

To learn more about Fastly's Media Shield, contact your account manager or email sales@fastly.com (mailto:sales@fastly.com) for more details.

§ About Fastly's On-the-Fly Packaging service (/guides/detailed-product-descriptions/about-fastlys-onthefly-packaging-service)

Fastly offers an "on-the-fly," dynamic, video-on-demand content packaging service. Rather than requiring you to pre-package all protocols of a viewer-requested video, Fastly allows you to dynamically package video content in different HTTP streaming formats in real time, using source

files. That video content then becomes immediately available to viewers.

❗ **IMPORTANT:** Fastly's On-the-Fly Packager (OTFP) for On Demand Streaming service is an add-on service. Our Professional Services team will assist with configuration and testing. To enable OTFP and begin this process, contact your account manager or email sales@fastly.com (mailto:sales@fastly.com) for more details.

Supported on-the-fly packaging features

Fastly's OTFP service supports the following specific features:

Supported HTTP streaming formats and codecs

- **HDS, HLS, and MPEG-DASH packaging.** Fastly provides support for version 1 of the Adobe HTTP Dynamic Streaming (HDS) specification and support for the ISO/IEC 23009-1:2014 specification (https://www.iso.org/standard/65274.html) defining Dynamic Adaptive Streaming over HTTP (MPEG-DASH). We support all features included in up to version 3 (draft 6) of the HTTP Live Streaming (HLS) specification and popular features from later versions such as subtitle, trick play and media segmentation in fragmented MPEG-4 (fMP4) format (https://tools.ietf.org/html/draft-pantos-http-live-streaming#section-3.3) (per ISO/IEC 14996-12:2015 specification (https://www.iso.org/standard/68960.html)).
- **Standard codecs.** Fastly supports Advanced Video Coding (H.264/AVC/MPEG-4 Part 10) and High Efficiency Video Coding (H.265/HEVC) video codecs. Fastly also supports Advanced Audio Coding (AAC, AAC-LC, HE-AAC), Dolby Digital (AC-3) and MPEG-1 Audio Layer III (MP3) audio codecs.
- **Source video container format.** Fastly supports the Progressive MP4 specification (specifically the .mp4, unencrypted .mov, and audio-only .m4a extensions) as source container format for packaging into all supported HTTP streaming formats.

Accessibility and user experience

- **HLS multi-language subtitles and closed captions.** Fastly provides support for both in-band (EIA-608 (https://en.wikipedia.org/wiki/EIA-608) and CEA-708 (https://en.wikipedia.org/wiki/CEA-708)) and out-of-band (Web Video Text Tracks or WebVTT (https://w3c.github.io/webvtt/)) subtitle and closed caption delivery.
- **HLS trick play.** Fastly supports trick play (also called trick mode), a feature that displays video scenes during fast-forwarding and rewinding. The HLS Authoring Specification (https://developer.apple.com/library/content/documentation/General/Reference/HLSAuthoring CH2-SW1) requires this feature for distributing video on the Apple TV.

Content protection

- **Media encryption.** Fastly can encrypt videos packaged into HLS (supports both Envelope/AES-128 and SAMPLE-AES (https://developer.apple.com/library/content/documentation/AudioVideo/Conceptual/HLS_Sa methods) and MPEG-DASH (ISO/IEC 23001-7, a common encryption in ISO base media file format file) streaming formats by generating a unique content encryption key for each video, enabling secure video delivery to viewers.
- **Multi-DRM.** Fastly can support multiple Digital Rights Management (DRM) technologies including Apple FairPlay (https://developer.apple.com/streaming/fps/) for HLS and Microsoft PlayReady (https://www.microsoft.com/playready/), Google Widevine (http://www.widevine.com/) and Marlin DRM (http://www.marlin-community.com/) for MPEG-DASH streaming formats. OTFP is integrated with Multi-DRM service providers that are responsible for content rights management and DRM license delivery.

Dynamic Ad Insertion (DAI) readiness

- **HLS timed metadata injection.** Fastly supports HLS time-based metadata (https://developer.apple.com/library/content/documentation/AudioVideo/Conceptual/HTTP_L which allows you embed custom metadata or ad markers about a stream into video segments at specified time instances in ID3v2 format.
- **Content preconditioning.** Fastly can segment video at the intended break points, such as for ad markers via HLS and MPEG-DASH protocols. Fastly can also add any third-party service-specific cues or metadata into video manifests at those break points to implement server or client-side ad stitching.

Live-to-VOD transition

- **Clip creation (also known as "timeline trimming").** Fastly supports clip creation features for all supported packaging formats, allowing you to deliver sections of video without segmenting a longer, archived video.

Fastly also provides the following features as part of standard content delivery network services:

- Token-based authentication (/guides/tutorials/enabling-token-authentication) for increased response time by placing validation at the edge
- Geolocation (/guides/vcl/geolocation-related-vcl-features) and device detection (/guides/vcl/delivering-different-content-to-different-devices) for content targeting
- Edge dictionaries (/guides/edge-dictionaries/) for real-time business rules and decision making at the edge
- Remote log streaming (/guides/streaming-logs/) for data aggregation and viewer diagnostics
- Transport Layer Security (TLS) (/guides/securing-communications/) for secure communications delivery

How the on-the-fly packaging service works

Fastly's OTFP service gets configured between our caching network and your origin storage (e.g., Amazon S3, Google Cloud Storage, or Rackspace Cloud Files).



When users request manifests or video segments, those requests initially come to Fastly caches instead of going to your origin storage. Fastly's edge caches deliver those objects if they are available and valid. If the objects don't already exist in the edge caches, the requests will be passed on to a designated shield cache (/guides/performance-tuning/shielding) to be delivered instead as long as the objects are available and valid. If neither the edge caches nor the shield cache can deliver the objects, the requests for those objects will go directly to and be fulfilled by the OTFP service which acts as an origin for Fastly's cache nodes.

The OTFP service will make the necessary request to your origin storage to fulfill the original request from the user. The OTFP service also maintains a small, local, in-memory cache for video metadata indexes. These indexes are created using mp4 moov atom (or movie atom) that provide information about the video file such as its timescale, duration, audio and video codec information, and video resolution (among other characteristics).

For adaptive bitrate playback (/guides/on-the-fly-packaging/adaptive-bitrate-playback-url-guidelines), the OTFP service will cache indexes of each quality level requested. If a user requests a manifest, OTFP will look for the corresponding indexes and, if it is available and valid, OTFP will generate the manifest and deliver it to the user. Otherwise, OTFP will fetch the moov atom from origin storage to generate the corresponding index. If a user requests video segments, OTFP will look for the corresponding audio and video sample entries in the cached index, download those samples from origin storage, and package them in the format requested.

---

§ About Fastly's Real-Time Log Streaming features (/guides/detailed-product-descriptions/about-fastlys-realtime-log-streaming-features)

To help you tune the performance of your Fastly services, we support real-time log streaming of data that passes through Fastly. We support a number of protocols that allow you to stream logs to a variety of locations, including third-party services, for storage and analysis.

Supported protocols and logging providers

Fastly supports a variety of syslog-compatible logging providers, such as Sumo Logic (/guides/streaming-logs/log-streaming-sumologic), Papertrail (/guides/streaming-logs/log-streaming-papertrail), and Logentries (/guides/streaming-logs/log-streaming-logentries). In addition, we provide a syslog endpoint (/guides/streaming-logs/log-streaming-syslog) specifically for sending log files to other syslog-based software (for example, to Logstash (https://www.elastic.co/products/logstash), part of the ELK stack, which supports input via syslog (https://www.elastic.co/guide/en/logstash/current/plugins-inputs-syslog.html)).

We also support other methods of sending logs besides the syslog protocol. We allow pushing of log files to Amazon S3 (/guides/streaming-logs/log-streaming-amazon-s3) buckets as well as any S3-compatible providers (such as DreamHost's DreamObjects). And we support FTP uploading (/guides/streaming-logs/log-streaming-ftp).

Find our supported logging endpoints in our list of streaming log guides (/guides/streaming-logs/). If the logging endpoint you're looking for isn't here, contact support@fastly.com (mailto:support@fastly.com) for suggestions on another endpoint that might provide the same functionality.

Supported log streaming features

Fastly's real-time log streaming supports the following specific features:

- **TLS support.** Fastly allows logging configuration information to be sent over TLS (Transport Layer Security) for certain endpoints. This means that logging information can be encrypted while in transit, which allows you to send potentially sensitive information to log files without exposing data.
- **Encryption.** Fastly allows you to encrypt log files (/guides/streaming-logs/encrypting-logs) for certain endpoints before they are written to disk. We encrypt files using OpenPGP (Pretty Good Privacy) (https://en.wikipedia.org/wiki/Pretty_Good_Privacy). For our Amazon S3 endpoint (/guides/streaming-logs/log-streaming-amazon-s3) in particular, we also support server-side encryption (http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html).
- **Customized log formats.** Fastly allows you to change the format (/guides/streaming-logs/custom-log-formats) of your logs by providing variables compatible with the Apache Common Log Format (https://httpd.apache.org/docs/trunk/logs.html#common) (NCSA Common log format).

- **Log file locations.** Fastly provides two different ways for you to change where your log files are written (/guides/streaming-logs/changing-where-log-files-are-written) for certain endpoints. You can change a log file's timestamp format (for example, if you wanted to remove characters from the log file name) and you can control the specific path to which those files are written.
- **Whitelisting.** Fastly's publicly available list of IP ranges (/guides/securing-communications/accessing-fastlys-ip-ranges) allow you to enable Fastly-only access to your logging servers through your firewall.

How Real-Time Log Streaming works

Varnish sends all streaming log records to a log aggregator, which streams them in near-real-time to the logging endpoint you configure (/guides/streaming-logs/setting-up-remote-log-streaming).

---

§ About Fastly's Video Caching service (/guides/detailed-product-descriptions/about-fastlys-video-caching-service)

For customers with their own video packaging infrastructure, Fastly can act as a globally distributed HTTP cache to improve quality of service and increase viewer capacity. When a manifest or video segment is requested by a customer's player, a Fastly edge or shield POP will pull the requested content from the customer's origin media server. Subsequent requests for that content will be served from Fastly's cache servers instead of the customer's origin (read *How Fastly's CDN Services Work* (/guides/basic-concepts/how-fastlys-cdn-service-works) for more information).

Fastly can cache and deliver any HTTP based media streaming protocol including:

- HTTP Live Streaming (HLS),
- HTTP Dynamic Streaming (HDS),
- HTTP Smooth Streaming (HSS), and
- Dynamic Adaptive Streaming over HTTP (MPEG-DASH).

In addition, both live and on-demand streams are supported.

---

§ Always-on DDoS mitigation (/guides/detailed-product-descriptions/always-on-ddos-mitigation)

Fastly's globally distributed network was built to absorb DDoS attacks. As part of Fastly's standard CDN services, all customers receive:

- **Access to origin shielding.** Fastly allows you to designate a specific point of presence (POP) to host cached content from your origin servers. This POP acts as a "shield (/guides/detailed-product-descriptions/about-fastlys-full-site-delivery-features#origin-shielding)" that protects those servers from every cache miss or pass through the Fastly network, reducing the load that directly reaches them.
- **Automatic resistance to availability attacks.** Before they're even processed by our caching infrastructure (/guides/basic-concepts/how-fastlys-cdn-service-works), we filter out Layer 3 and 4 attacks (e.g., Ping floods, ICMP floods, UDP abuse) as well as distributed reflection and amplification (DRDoS) attacks that rely on anonymity to abuse internet protocols (e.g., DNS and NTP).
- **Access to Fastly cache IP space.** Fastly provides an API endpoint to any customer who would like to know which IP addresses (/guides/securing-communications/accessing-fastlys-ip-ranges) our caches will use to send traffic from our CDN to your origin servers. We make this data available so you can update firewalls at your origin to ensure only our cache traffic can access your resources.
- **Custom DDoS filter creation abilities.** Using custom VCL (/guides/vcl/guide-to-vcl), we allow you to craft your own DDoS protection rules to protect your network from complex Layer 7 attacks. Once you identify signs of a potential DDoS attack, you can mix and match Fastly VCL with custom VCL (/guides/vcl/mixing-and-matching-fastly-vcl-with-custom-vcl) to construct filter configurations based on a variety of client and request criteria (e.g., headers, cookies, request path, client IP, geographic location) that block malicious requests before they hit your origin servers.

In addition to these standard DDoS protection services, Fastly offers a DDoS Protection and Mitigation Service (/guides/detailed-product-descriptions/ddos-protection-and-mitigation-service-and-sla). For more information about this or any of our advanced services, including their subscription costs, contact sales-ddos@fastly.com (mailto:sales-ddos@fastly.com).

§ Assurance Services (/guides/detailed-product-descriptions/assurance-services)

Subscribers who purchase Assurance Services will:

- have access to a library of third-party audit reports and certification attestations (most recent 12 months).
- have access to executive summary reports for penetration tests and network scans (most recent 12 months).
- have access to a library of security-related policies and procedures.
- have access to a library of executive summaries of annual risk assessments (most recent 12 months).
- have access to a library of historical Fastly Service Advisory (FSA) documents (most recent 12 months).
- be able to perform unlimited audits of Fastly's security (/guides/compliance/security-program) and technology compliance (/guides/compliance/technology-compliance) programs, subject to Subscriber's purchase of Professional Services (/guides/detailed-product-descriptions/professional-services). Audits require advance notice of at least 10 business days and shall be performed by Subscriber (or a mutually acceptable third party) according to standard audit practices.
- have the ability to be added as an Additional Insured on Fastly's General Commercial Liability Insurance for an additional fee.

Subscribers who wish to purchase Assurance Services must also purchase Gold or Platinum Support (/guides/detailed-product-descriptions/support-description-and-sla).

§ DDoS Protection and Mitigation Service and SLA (/guides/detailed-product-descriptions/ddos-protection-and-mitigation-service-and-sla)

Fastly offers DDoS Protection and Mitigation Service to customers with a sustained DDoS threat risk or with short term and seasonal events to protect. While the DDoS Protection and Mitigation Service cannot prevent or eliminate attacks or guarantee the uptime of your origin servers, it offers the following resources to assist you with mitigating the service and financial impacts of DDoS and related attacks.

Fastly's DDoS Protection and Mitigation Service includes:

- Immediate onboarding - We will work directly with you to immediately transition you to Fastly's CDN service if you're not already a customer.
- Emergency configuration and deployment support - We will actively work with you to configure your service map and provide an initial filter policy to immediately block an attack.
- Ongoing attack mitigation support - We will work directly with you to write custom VCL filters to deal with changing attacks or new attacks. We'll also isolate malicious traffic on your behalf.
- Incident response plan - We will create a plan that identifies how communication and escalation will occur between you and your staff and Fastly if an attack occurs. The plan will also describe mitigation and defense details such as any DDoS filters that we can insert into VCL prior to or during an attack.

Using our knowledge of attacks against our network and our customers, we analyze all DDoS Attack vectors using VCL statements, network filters, bulk traffic filtering through regional sinks, or a combination of these techniques.

The following table summarizes what is provided under our DDoS Protection and Mitigation Service:

Support offering	**Details**
Online self-service help	Unlimited access.
Availability for general inquiries	24/7.
Availability for incident reports	24/7.
Initial response times	Attack notification response within 15 minutes. Service onboarding beginning within 60 minutes of threat notification.
Overage Insurance	Included.
Access to Fastly IP Space	Included.
Email support	Available.
Phone and chat support	Toll-free telephone available 24/7/365. Dedicated chat channel available during Fastly Business Hours.
Emergency escalation	Available via email and phone support.

Technical support

The following section applies to all Subscribers of the DDoS Protection and Mitigation Service.

Definitions

- "Business Hours" are 8AM-6PM during a Business Day in California, New York, London, or Tokyo.
- "Business Days" are Monday through Friday, excluding any day that is simultaneously a US, UK, and Japanese national or banking holiday.
- A "DDoS Attack" is a Denial of Service (DoS) event (including Distributed Denial of Service (DDoS) or Distributed Reflection Amplification Denial of Service (DRDoS) attacks) that includes both an increase of unwanted traffic beyond two (2) times the average traffic of any Fastly Service for the preceding two (2) month period and a simultaneous increase in error responses from origin sites configured for any Fastly service. Fastly captures and analyzes suspected or actual DDoS Attack traffic to improve and protect its services.
- A "Fastly IP Space" is a published API endpoint (https://api.fastly.com/public-ip-list) that allows Subscribers to download an updated list of all Fastly IPs globally and can be used to filter traffic and control communication between Fastly's caches and a Subscriber's origin. Fastly provides the Fastly IP Space to Subscribers in order to ensure known communication between the Fastly cache nodes and a Subscriber's origin datacenter.
- "Fastly Control" means elements entirely under Fastly's control and not a consequence of (a) a Subscriber's hardware or software failures, (b) a Subscriber's or end user's connectivity issues, (c) Subscriber operator errors, (d) Subscriber traffic amounts that exceed a Subscriber's Permitted Utilization as defined in the Service Availability SLA (/guides/detailed-product-descriptions/service-availability-sla), (e) corrupted Subscriber content, (f) acts of god (any) or war, or earthquakes, or terrorist actions.

Subscriber responsibilities

As a Subscriber, you:

- must identify and maintain two points of contact to be used during an attack to communicate status, issues, and coordinate with Fastly to successfully protect services.
- must use common best practices for DDoS Attack defense including:
  - using updated white and black lists in the Fastly IP Space at the origin datacenter to protect against attack traffic bypassing Fastly's infrastructure.
  - limiting or eliminating your origin IP addresses from Domain Name System (DNS) records to avoid these addresses being used as attack targets.

- are responsible for using and configuring services according to the documentation available at https://docs.fastly.com (/guides/).

Support requests

Subscribers may make support requests by submitting a support ticket which will trigger a system-generated acknowledgement within minutes containing the ticket number and a direct link to the ticket.

DDoS Attack reports should include at least:

- a determination of the severity of the attack.
- the size of the attack threatened or previously observed.
- the type and vector of attack traffic seen or threatened.
- any duration of previous attacks and vector behavior including major source IP addresses.
- attack history for the last 24 months.
- threat specifics including all details of any attacks that the protected services or sites have experienced in the past.

Communications and channels of support

Support tickets

Create support tickets by sending an email to support@fastly.com (mailto:support@fastly.com) or calling our dedicated phone line. Filed tickets trigger Fastly's promised response time.

Tickets for communication between Fastly support engineers and a Subscriber's personnel are tracked using a ticketing application, which maintains a time-stamped transcript of communications, and sends emails to Subscriber and Fastly staff as tickets are updated.

Phone support

Subscribers to the DDoS Protection and Mitigation Service receive a dedicated phone number to contact Fastly support engineers. Fastly personnel can also establish audio and video conferencing (free app or browser plug-in required) for real-time voice and video communications.

Chat

To facilitate real-time communication, Subscribers to the DDoS Protection and Mitigation Service receive a dedicated chat channel for real-time communications during Business Hours or as needed by Fastly personnel. Though subject to change, Fastly's current chat provider is Slack (www.slack.com (https://slack.com/)).

Attack traffic

Response time

Fastly commits to responding to DDoS Attack notifications from Subscribers within 15 minutes of notice and, as applicable, will begin on-boarding Subscribers to the DDoS Protection and Mitigation Service within 60 minutes of a DDoS Attack notification.

Related Invoice Credits

Fastly will waive all bandwidth and request charges associated with DDoS Attack traffic and will provide Invoice Credits or adjustments for the same.

Attack traffic credit terms

Subscribers must submit claims for waiver of DDoS Attack-related charges to billing@fastly.com (mailto:billing@fastly.com) within 30 days of the DDoS Attack.

DDoS Mitigation response SLA

If, during a DDoS Attack on a Subscriber with DDoS Protection and Mitigation Service, there is a material delay in response time and the cause of the delay is within Fastly's control, a one-time credit of $500 per incident will be credited to that Subscriber's account.

SLA credit terms

- Requests for Invoice Credits must be made within 30 days of the DDoS Attack that triggered the service credit.
- All requests for Invoice Credits must be made to billing@fastly.com (mailto:billing@fastly.com).
- In no event shall Invoice Credits exceed the fee for the DDoS Protection and Mitigation Service payable by a Subscriber for the month in which the Invoice Credits accrued.
- A pending Invoice Credit does not release a Subscriber from the Subscriber's obligation to pay Fastly's submitted invoices in full when due.
- Invoice Credits will be applied to the invoice within the month the credits were incurred.

Termination for SLA

For a Subscriber of the DDoS Protection and Mitigation Service with a Termed Contract, if in any three-month period where three (3) or more support response time objectives are not met and the failure to meet the objectives materially adversely impacted the Subscriber, the Subscriber will

have 30 days to terminate the DDoS Protection and Mitigation Service subscription following the third response failure. Subscribers must notify Fastly of their intention to terminate the DDoS Protection and Mitigation Service subscription within 30 days of the triggering event.

§ HIPAA and caching PHI (/guides/detailed-product-descriptions/hipaa-and-caching-phi)

You can configure the Fastly CDN service to cache and transmit protected health information (PHI) in keeping with Health Information Portability and Accountability Act (HIPAA) security requirements. Use the following features to ensure secure handling of cache data that contains PHI:

- Configure frontend (/guides/detailed-product-descriptions/tls-service-options) and backend (/guides/basic-configuration/connecting-to-origins#setting-the-tls-hostname) TLS to encrypt transmitted data from your origin to your end users.
- Add the `beresp.hipaa` variable (/guides/vcl/miscellaneous-VCL-extensions) to objects containing PHI to keep that data out of non-volatile disk storage at the edge.

Contact sales@fastly.com (mailto:sales@fastly.com) for more information on how to enable the `beresp.hipaa` feature for your account. For accounts that have this feature enabled, Fastly will enter into a HIPAA business associate agreement (BAA) as an addendum to our terms of service (https://www.fastly.com/terms).

NOTE: Fastly's security and technology compliance program includes safeguards for the entire Fastly CDN service, independent of using the `beresp.hipaa` variable. The Fastly security program (/guides/compliance/security-program) and technology compliance (/guides/compliance/technology-compliance) guides provide more information about these safeguards.

§ Legacy Premium Support and SLA (/guides/detailed-product-descriptions/legacy-premium-support-and-sla)

ℹ **NOTE:** Fastly maintains support for its original Premium Support plan. For more information about our current Gold and Platinum support plans (/guides/detailed-product-descriptions/support-description-and-sla) or for information about our Professional Services packages (/guides/detailed-product-descriptions/professional-services), contact sales@fastly.com (mailto:sales@fastly.com).

Legacy Premium Support description and SLA

Support availability and response times vary depending on the type of account you have and the level of support you have purchased. The following table summarizes those offerings:

Offering	**Unpaid Account**	**Month-to-Month Account**	**Termed Contact**	**Premium Support**
Online Forums	Yes	Yes	Yes	Yes
Email Support Response Time Commitment	Best Effort	Best Effort	Next Business Day	Severity 1 Incidents: 15 minutes*. All Others: Next Business Day
Severe Incident Response Email Address	No	No	No	Yes
Support SLA	None	None	Termination Option	Invoice Credits + Termination Option

Technical support

The following section applies to all Subscribers.

Definitions

- **"Business Hours"** are 8AM-6PM Monday through Friday, Pacific Time.
- **"Business Days"** are Monday through Friday excluding US and UK national and banking holidays.
- An **"Incident"** is an occurrence during which an end user's use of Subscriber's services is adversely impacted.
- A **"Severity 1 Incident"** is an Incident resulting in a major service outage requiring Subscriber to redirect all traffic from Fastly to another CDN.
- A **"Severity 2 Incident"** is an Incident resulting in minor or intermittent outage not requiring Subscriber to redirect traffic to another CDN.

- **"Fastly Control"** means elements entirely under Fastly's control and not a consequence of (a) Subscriber hardware or software failures, (b) Subscriber or end user connectivity issues, (c) Subscriber operator errors, (d) Subscriber traffic amounts that exceed Subscriber's Permitted Utilization as defined in the Terms and Conditions, (e) corrupted Subscriber content, (f) acts of god (any) or war, or earthquakes, or terrorist actions.

Subscriber responsibilities

Subscriber is responsible for using and configuring services according to the Documentation available at https://docs.fastly.com (/).

Support requests

Subscribers submit support requests by sending email to support@fastly.com (mailto:support@fastly.com). Subscribers receive a system-generated response within minutes containing the ticket number and a direct link to the ticket.

Incident reports should include at the least the following:

- Services not responding to end user requests.
- Services incorrectly sending end users error condition messages.
- Services sending incorrect or partial content to end-users.

Incident reports should include all relevant information, such as:

- Subscriber's determination of the Severity Level of the Incident,
- Subscriber hardware failures,
- Subscriber operator errors,
- Services configuration errors made by Subscriber employees,
- Potential Excess Utilization (as defined in the Terms of Use or master services agreement),
- Corrupted Subscriber content,
- DDOS attacks, and
- Relevant *force majeure* acts such as extreme weather, earthquakes, strikes or terrorist actions.

Communications

Communications between Fastly support engineers and Subscriber staff are conducted using the ticketing application, which maintains a time-stamped transcript of all communications, and sends emails to Subscriber and Fastly staff as tickets are updated.

Response time

Fastly shall use best efforts to respond in a timely fashion.

Termed contracts

The following applies to any Subscriber that has a contract with a term and a minimum commitment.

Response times

Fastly commits to acknowledging receipt of a support ticket within the next business day following submission of a support request.

Termination

In any three-month period where three (3) or more support Response Time objectives are not met and the failure to meet the objectives materially adversely impacted Subscriber, Subscriber shall have thirty (30) days to terminate their subscription agreement following the third failure.

Premium Support

The following applies to Subscribers who have purchased Premium Support.

Incident reporting

Severity 1 Incidents: Fastly will provide Subscriber an Incident Support Email address for Subscriber to report Incidents. Subscriber should report Incidents promptly using the Incident Support email.

Severity 2 Incidents: Subscriber should report Severity 2 Incidents by submitting a Support Request.

Response times

Fastly will respond to the report of an Incident by troubleshooting the cause(s) of the Incident and resolve them if caused by factors within Fastly's control, or provide information to those who can resolve the factors if the factors are within others' control, as follows:

For a Severity 1 Incident:

- Fastly support staff will acknowledge receipt of the email within 15 minutes.
- Fastly will start actively troubleshooting within 30 minutes of receipt of the email.
- Fastly will perform its tasks on a 24/7 basis.
- Fastly and Subscriber will immediately communicate upon learning new information that may be useful in troubleshooting the Incident, and status updates between Fastly and Subscriber staff will take place no less frequently than every 30 minutes for the first two hours, and no less frequently than every hour thereafter.
- Fastly staff will work until (a) the Incident is resolved or (b) the Incident is believed to be outside of Fastly's control.

For a Severity 2 Incident:

- During Business Hours, Fastly support staff will acknowledge receipt of the email within two hours or within two hours of the start of the next business day if the Incident does not come in during a Business Day.
- Fastly engineers will begin actively troubleshooting within one business day, will work on the Incident during Business Hours, and will provide status updates to Subscriber daily on each subsequent Business Day.

Charges for Incident Response

For Severity 1 Incidents caused by factors within Subscriber's control, a flat fee of $1500 will be assessed, and any time spent beyond three hours will be invoiced at Subscriber's undiscounted Professional Services rates.

For Severity 2 Incidents caused by factors within Subscriber's control, Subscriber will be invoiced at Subscriber's undiscounted Professional Services Rates.

For all Incidents:

- If the Incident-causing factors are within Fastly's control, there will be no hourly charges for Fastly engineering staff time.
- If the factors are within Subscriber's control, Subscriber agrees to pay Fastly its hourly charges for Fastly engineering staff time. If it appears likely the factors are within Subscriber's, Subscriber may tell Fastly staff to stop working on troubleshooting the Incident (thereby stopping the hourly charges from being incurred). Subscriber agrees to tell Fastly to stop working on an Incident via an email sent to Fastly's Incident Support email address. The timestamp on the email will be the time charges cease to be incurred.

Support Invoice Credits

In the event a Severity 1 Incident occurs, Subscriber has purchased Premium Support, the cause of the Incident is within Fastly's control, and any of the communication or response timeframes are materially not met, a one-time credit of $500 per Incident will be credited to Subscriber's account.

Credit Terms:

- Requests for Invoice Credits must be made within 30 days of the Incident which triggered the service credit.
- In no event shall Invoice Credits exceed the invoice value of the month in which they are accrued.
- A pending credit does not release Subscriber from its obligation to pay Fastly's submitted invoices in full when due.
- Credits will be applied to the invoice two months following the month an invoice credit was incurred.

Legacy Service availability SLA

Support availability and response times vary depending on the type of account (/guides/account-types-and-billing/accounts-and-pricing-plans) you have and the level of support (/guides/detailed-product-descriptions/support-description-and-sla) you have purchased.

Agreement Type	Unpaid Account	Month-to-Month Account	Termed Contract	Premium Support
Service Level Agreement	None	None	Termination Option	Invoice Credits + Termination Option

Definitions

"Degraded Performance" for the Services means the Services are experiencing Error Conditions that are (1) caused by issues under Fastly Control, (2) observable or reproducible by Subscriber or Fastly, (3) requiring Subscriber to redirect traffic off the Services. Degraded Performance does not include any reduction on availability of the Application User Interface or API due to planned maintenance.

"Error Condition" means the Services are (1) not responding to end user requests, (2) incorrectly sending end users error condition messages or (3) sending incorrect partial content to end users and these conditions are observable or reproducible by Subscriber or Fastly.

"Fastly Control" means elements entirely under Fastly's control and not a consequence of (a) Subscriber hardware or software failures, (b) Subscriber or end user connectivity issues, (c) Subscriber operator errors, (d) Subscriber traffic amounts that exceed Subscriber's Permitted Utilization, (e) corrupted Subscriber content, (f) acts of god (any) or war, or earthquakes, or terrorist actions.

Termination

Any Subscriber that has a contract with a term and a minimum commitment shall have thirty (30) days to terminate their subscription agreement if the Services experience Degraded Performance (a) for longer than 7.2 hours in any one month, or (b) for longer than 43.8 minutes each month in any three contiguous months. Subscriber shall have thirty (30) days to terminate their contract following the third failure.

Availability of invoice credits

Subscribers who purchase Premium Support shall be entitled to Invoice Credits according to the following table.

Availability Percent	Period of Degraded Performance	Monthly Credit Percent
Below 100% - 99.99%	Up to 4.32 minutes	1%
99.99% – 99.9%	Up to 43.8 minutes	5%

Availability Percent	Period of Degraded Performance	Monthly Credit Percent
99.89% – 99.0%	Up to 7.2 hours	10%
98.99% - 98.0%	Up to 14.4 hours	25%
Below 97.99%	Greater than 864 minutes	50%

Invoice Credits for unavailability will accrue on a monthly basis. The Credit Amount for a month is equal to the monthly usage charge multiplied by Monthly Credit Percent.

Credit terms

- Requests for Invoice Credits for Degraded Performance must be made within 30 days of the period of Degraded Performance.
- The maximum amount of any credit is the Invoice Amount for the month the Degraded Performance occurred.
- A pending credit does not release Subscriber from its obligation to pay Fastly's submitted invoices in full when due.
- Credits will be applied to the Invoice two months following the month an invoice credit was incurred.

---

§ PCI-compliant caching (/guides/detailed-product-descriptions/pci-compliant-caching)

We have designed Fastly's core CDN service with Payment Card Industry Data Security Standard (PCI DSS) compliance in mind. With proper authorization on your account, you can use Fastly's `beresp.pci` VCL variable (/guides/vcl/miscellaneous-VCL-extensions) to automatically cache content in a manner that satisfies PCI DSS requirements.

Adding the `beresp.pci` variable to an object prevents writing of that object to non-volatile disk storage on the edge. Combined with frontend (/guides/detailed-product-descriptions/tls-service-options) and backend TLS (/guides/basic-configuration/connecting-to-origins#setting-the-tls-hostname), this feature allows you to cache and transmit flagged content through the Fastly network in compliance with our PCI certification.

Contact sales-ecommerce@fastly.com (mailto:sales-ecommerce@fastly.com) for more information on how to enable this feature for your account.

ℹ **NOTE:** Fastly's security and technology compliance program includes safeguards for the entire Fastly CDN Service, independent of using the `beresp.pci` variable. The Fastly security program (/guides/compliance/security-program) and technology compliance (/guides/compliance/technology-compliance) guides provide more information about these safeguards.

§ Professional Services (/guides/detailed-product-descriptions/professional-services)

Fastly offers a range of Professional Services to help you begin using Fastly services. Choose between Service Implementation, Service Management, or Consulting Engagement Services, depending on your needs. For more information about any of our Professional Services packages, contact sales@fastly.com (mailto:sales@fastly.com).

Service Implementation

How it works

Fastly Professional Services staff will personally guide you through the following stages:

- **Planning:** Professional Services staff help you with requirements gathering, solution design, documentation and resource allocation.
- **Implementation:** Professional Services staff help you with configuration of Fastly services and custom VCL development. They provide best-practice consulting for configuration of your origins.
- **Testing:** Professional Services staff help you validate configurations and set up testing.
- **Go-Live:** Professional Services staff monitor and address issues during final production testing and deployment.

Implementation, Testing, and Go-Live may involve some iterative cycles depending on the complexity of your configuration.

Implementation options

Some common implementation options we offer include:

- Initial setup and configuration
- End-to-end encryption setup

- Fine-tuning cache times
- Custom header (/guides/basic-configuration/adding-or-modifying-headers-on-http-requests-and-responses) logic
- Dynamic content delivery optimization
- Multi-tiered caching setup
- Lightweight web page hosting
- Custom purging (/guides/purging/) and event-driven content management
- Geographic or localization detection (/guides/performance-tuning/changing-origins-based-on-user-location)
- Edge logic and device detection (/guides/vcl/delivering-different-content-to-different-devices)
- Stale content (/guides/performance-tuning/serving-stale-content) configuration and origin outage handling
- Edge authentication and authorization
- ESI (edge side includes (/guides/performance-tuning/using-edge-side-includes))
- Streaming (/guides/detailed-product-descriptions/about-fastlys-video-caching-service) and video packaging (/guides/detailed-product-descriptions/about-fastlys-onthefly-packaging-service)
- Site performance analysis
- Managed vendor migration

Fastly offers two Service Implementation packages:

- **Standard:** Basic implementation for Fastly customers with simple content configurations.
- **Enterprise:** Advanced implementation for Fastly customers with complex, custom configurations.

Service Management

For customers who require ongoing configuration and technical assistance, Fastly offers Service Management that provide professional services to you and your staff on an as-needed basis. These hours may be used to supplement your existing Support Plan or Service Implementation.

Some common activities you may need assistance with:

- Site performance analysis
- Varnish and VCL (/guides/vcl/guide-to-vcl) training
- Service (/guides/basic-setup/working-with-services) configuration

- End-to-end encryption setup
- Cache time fine-tuning
- Custom header logic creation
- Dynamic content delivery optimization
- Multi-tiered caching setup
- Lightweight web page hosting
- Custom purging and event-driven content management
- Geographic or localization detection
- Edge logic and device detection
- Stale content configuration and origin outage handling
- Edge authentication
- ESI (edge side includes) configuration
- Streaming and video packaging

Consulting Engagement Services

For customers who require in-house expertise or dedicated resources, Fastly's Support Engineers are available to provide a range of more technical professional services, including:

- Technical advisory services
- Translating configurations to VCL
- Optimization of website performance
- On-site Varnish and VCL training
- Non-Fastly related performance tuning
- Adapting Fastly features to a particular customer use case

---

§ Service availability SLA (/guides/detailed-product-descriptions/service-availability-sla)

Support availability and response times vary depending on the type of account (/guides/account-types-and-billing/accounts-and-pricing-plans) you have and the level of support (/guides/detailed-product-descriptions/support-description-and-sla) you have purchased.

Agreement Type	Unpaid Account	Month-to-Month Account	Termed Contract	Gold & Platinum Support
Service Level Agreement	None	None	Termination Option	Invoice Credits + Termination Option

Definitions

"Degraded Performance" means the Services are experiencing Error Conditions that are (1) caused by issues under Fastly Control, (2) observable or reproducible by Subscriber or Fastly, (3) requiring Subscriber to redirect traffic off the Services. Degraded Performance does not include any reduction on availability of the Application User Interface or API due to maintenance.

"Error Condition" means the Services are (1) not responding to end user requests, (2) incorrectly sending end users error condition messages or (3) sending incorrect partial content to end users and these conditions are observable or reproducible by Subscriber or Fastly.

"Fastly Control" means elements entirely under Fastly's control and not a consequence of (a) Subscriber hardware or software failures, (b) Subscriber or end user connectivity issues, (c) Subscriber operator errors, (d) a Utilization spike (see below), (e) corrupted Subscriber content, (f) acts of god (any) or war, or earthquakes, or terrorist actions.

Termination

Any Subscriber that has a contract with a term and a minimum commitment shall have thirty (30) days to terminate their subscription agreement following (1) a period of Degraded Performance longer than 7.2 hours in any one month, or (b) three contiguous months that have periods of Degraded performance longer than 43.8 minutes each.

Availability invoice credits

Subscribers who purchase Gold or Platinum Support shall be entitled to Invoice Credits according to the following table.

Availability Percent	Period of Degraded Performance	Monthly Credit Percent
Below 100% - 99.99%	Up to 4.32 minutes	1%
99.99% – 99.9%	Up to 43.8 minutes	5%
99.89% – 99.0%	Up to 7.2 hours	10%
98.99% - 98.0%	Up to 14.4 hours	25%
Below 97.99%	Greater than 864 minutes	50%

Invoice Credits for unavailability will accrue on a monthly basis. The Credit Amount for a month is equal to the monthly usage charge multiplied by Monthly Credit Percent.

Credit terms

- Requests for Invoice Credits for Degraded Performance must be made within 30 days of the period of Degraded Performance.
- The maximum amount of any credit is the Invoice Amount for the month the Degraded Performance occurred.
- A pending credit does not release Subscriber from its obligation to pay Fastly's submitted invoices in full when due.
- Credits will be applied to the Invoice two months following the month an invoice credit was incurred.

Utilization Spikes

Subscriber's bandwidth utilization, measured in megabits per second, will be sampled every five (5) minutes on a region-by-region basis each month (the "**Samples**"). Subscriber's "**Average Utilization**" for a region in a month will be the average of the Samples. Subscriber's "**Peak Utilization**" for a region in a month will be calculated by the 95th percentile method, according to which the Samples will then be ordered from highest to lowest, and the highest five percent (5%) of Samples will be discarded and the remaining highest Sample will be Subscriber's Peak Utilization for the region in that month. Subscriber's "**Permitted Utilization**" in a month for a region will be five (5) times Subscriber's Average Utilization in that month for that region. A "**Utilization Spike**" will occur if Subscriber's Peak Utilization exceeds its Permitted Utilization in a region. Utilization Spikes may interfere with or disrupt the integrity or performance of the Services. Subscribers should contact Support in advance of any planned utilization spike and respond immediately to any communications from Fastly regarding an actual or suspected Utilization Spike.

---

§ Support description and SLA (/guides/detailed-product-descriptions/support-description-and-sla)

Support availability and response times vary depending on the type of account you have and the level of support you have purchased. The following table summarizes those offerings:

Support Offering	**Standard Support**	**Gold Support**	**Platinum Support**

Support Offering	Standard Support	Gold Support	Platinum Support
Online Self-Service Help	Unlimited access.	Unlimited access.	Unlimited access.
Availability for General Inquiries	Business hours.	Business hours.	24/7/365.
Availability for Incident Reports	Business hours, including weekends & holidays.	24/7/365.	24/7/365.
Initial Response Times	By the next business day.	Severity 1 Incidents within 2 hours. Severity 2 Incidents within same day. All other Incidents by the next business day.	Severity 1 Incidents within 15 minutes. Severity 2 Incidents within 2 hours. All other Incidents by the next business day.
Email support	Available.	Available, with priority over Standard Support.	Available, with priority over Standard and Gold Support.
Phone and chat support	Not available.	Not available.	Toll-free telephone available 24/7/365. Dedicated chat channel available during Fastly business hours.
Emergency Escalation	Not available.	Not available.	Available via email and phone.
Designated Customer Support Engineer	Not available.	Not available.	Available for large accounts on case-by-case basis.
Termination Option	Not available for unpaid and month-to-month customers. Only included for termed contracts.	Available with invoice credits.	Available with invoice credits.

Technical support

The following section applies to all subscribers.

Definitions

- **"Business Hours"** are 8AM-6PM during a Business Day in California, New York, London, or Tokyo.
- **"Business Days"** are Monday through Friday, excluding any day that is simultaneously a US, UK, and Japanese national or banking holiday.
- An **"Incident"** is an occurrence during which end users' use of Subscriber's services is adversely impacted.
- A **"Severity 1 Incident"** is an incident resulting in a major service outage requiring Subscriber to redirect all traffic from Fastly to another CDN.
- A **"Severity 2 Incident"** is an incident resulting in minor or intermittent outage not requiring Subscriber to redirect traffic to another CDN.
- **"Fastly Control"** means elements entirely under Fastly's control and not a consequence of (a) a Subscriber's hardware or software failures, (b) a Subscriber's or end user's connectivity issues, (c) Subscriber operator errors, (d) Subscriber traffic amounts that exceed a Subscriber's Permitted Utilization as defined in the Terms and Conditions, (e) corrupted Subscriber content, (f) acts of god (any) or war, or earthquakes, or terrorist actions.

Subscriber responsibilities

Subscriber is responsible using and configuring services according to the Documentation available at https://docs.fastly.com (/).

Support requests

Subscribers submit support requests by sending email to support@fastly.com (mailto:support@fastly.com). Subscribers receive a system-generated response within minutes containing the ticket number and a direct link to the ticket.

Incident reports should include at the least the following:

- Services are not responding to end user requests.
- Services incorrectly send end users error condition messages.
- Services send incorrect or partial content to end users.

Incident reports should include all relevant information such as:

- Subscriber's determination of the Severity Level of the incident,
- Subscriber hardware failures,
- Subscriber operator errors,

- Services configuration errors made by Subscriber employees,
- A potential Utilization Spike (see the Service Availability SLA (/guides/detailed-product-descriptions/service-availability-sla)),
- Corrupted Subscriber content,
- DDOS attacks, and
- Relevant *force majeure* acts such as extreme weather, earthquakes, strikes or terrorist actions.

Communications

Tickets

Communications between Fastly support engineers and Subscriber personnel are conducted using the ticketing application, which maintains a time-stamped transcript of communications, and sends emails to Subscriber and Fastly staff as tickets are updated.

Chat

Subscribers to Platinum Support receive a dedicated chat channel for real-time communications during Business Hours. Though subject to change, Fastly's current chat provider is Slack (www.slack.com (https://slack.com/)).

Phone support

Subscribers to Platinum Support receive a dedicated phone number to contact Fastly support engineers. Fastly personnel can also establish audio and video conferencing (free app or browser plug-in required) for real-time voice and video communications.

Response time

Fastly shall use best efforts to respond in a timely fashion.

Termed contracts

The following applies to any subscriber that has a contract with a term and a minimum commitment.

Response times

Fastly commits to acknowledging receipt of a support ticket within the next Business Day following submission of a support request by a Subscriber with a Termed Contract.

Termination

In any three-month period where three (3) or more support Response Time objectives are not met and the failure to meet the objectives materially adversely impacted Subscriber, Subscribers with a Termed Contract, Gold Support or Platinum Support shall have thirty (30) days to terminate their subscription agreement following the third failure.

Incident response times

Incident reporting

Severity 1 Incidents: Fastly will provide Subscriber an Incident Support Email address for Subscriber to report Incidents. Subscriber should report Incidents promptly using the Incident Support email.

Severity 2 Incidents: Subscriber should report Severity 2 Incidents by submitting a Support Request.

Platinum Support

Fastly will respond to the report of an Incident by troubleshooting the cause(s) of the Incident and resolve them if caused by factors within Fastly's control, or provide information to those who can resolve the factors if the factors are within others' control, as follows:

For a Severity 1 Incident:

- Fastly support staff will acknowledge receipt of the email within 15 minutes.
- Fastly will start actively troubleshooting within 30 minutes of receipt of the email.
- Fastly will perform its tasks on a 24/7 basis.
- Fastly and Subscriber will immediately communicate upon learning new information that may be useful in troubleshooting the incident, and status updates between Fastly and Subscriber staff will take place no less frequently than every 30 minutes for the first two hours, and no less frequently than every hour thereafter.
- Fastly staff will work until (a) the incident is resolved or (b) the incident is believed to be outside of Fastly's control.

For a Severity 2 Incident:

- Fastly support staff will acknowledge receipt of the email within two hours.
- Fastly engineers will begin actively troubleshooting within the same day, will work on the Incident during the same day, and will provide status updates to Subscriber daily on each subsequent day.

Gold Support

Fastly will respond to the report of an Incident by troubleshooting the causes of the Incident and resolve them if caused by factors within Fastly's control, or provide information to those who can resolve the factors if the factors are within others' control, as follows:

For a Severity 1 Incident:

- Fastly support staff will acknowledge receipt of the email within two hours.
- Fastly engineers will begin actively troubleshooting within the same day, will work on the Incident during the same day, and will provide status updates to Subscriber daily on each subsequent day.
- Fastly staff will work until (a) the incident is resolved or (b) the incident is believed to be outside of Fastly's control.

For a Severity 2 Incident:

- Fastly support staff will acknowledge receipt of the email within the same day.
- Fastly engineers will begin actively troubleshooting within the same day, will work on the Incident during the same day or next day, and will provide status updates to Subscriber daily on each subsequent day.

For Severity 1 Incidents caused by factors within Subscriber's control, a flat fee of $1500 will be assessed, and any time spent beyond three hours will be invoiced at Subscriber's undiscounted Professional Services rates.

For Severity 2 Incidents caused by factors within Subscriber's control, Subscriber will be invoiced at Subscriber's undiscounted Professional Services Rates.

For all incidents:

- If the Incident-causing factors are within Fastly's control, there will be no hourly charges for Fastly engineering staff time.
- If the factors are within Subscriber's control, Subscriber agrees to pay Fastly its hourly charges for Fastly engineering staff time. If it appears likely the factors are within Subscriber's control, Subscriber may tell Fastly staff to stop working on troubleshooting the Incident (thereby stopping the hourly charges from being incurred). Subscriber agrees to tell Fastly to stop working on an Incident via an email sent to Fastly's Incident Support email address. The timestamp on the email will be the time charges cease to be incurred.

Support invoice credits

In the event a Severity 1 Incident occurs, Subscriber has purchased Gold or Platinum Support, the cause of the Incident is within Fastly's control, and any of the communication or response timeframes are materially not met, a one-time credit of $500 per incident will be credited to Subscriber's account.

Credit Terms:

- Requests for Invoice Credits must be made within 30 days of the incident which triggered the service credit.
- In no event shall Invoice Credits exceed the invoice value of the month in which they are accrued.
- A pending credit does not release Subscriber from its obligation to pay Fastly's submitted invoices in full when due.
- Credits will be applied to the invoice two months following the month an invoice credit was incurred.

NOTE: Fastly maintains support for its original Premium Support plan (/guides/detailed-product-descriptions/legacy-premium-support-and-sla). To convert your account to the current Gold and Platinum Support plans, contact sales@fastly.com (mailto:sales@fastly.com).

---

§ TLS service options (/guides/detailed-product-descriptions/tls-service-options)

The Fastly Transport Layer Security (TLS) service provides privacy and data security for your service. You can request different TLS service options for your account.

IMPORTANT: By June 30, 2018, all customers, including those on dedicated endpoints, must have converted to TLS-1.2 (https://www.fastly.com/blog/phase-two-our-tls-10-and-11-deprecation-plan). Due to the PCI Security Standards Council mandate (https://www.pcisecuritystandards.org/documents/Migrating_from_SSL_Early_TLS_Information%
older TLS implementations will no longer be supported on Fastly infrastructure on shared or dedicated endpoints after this date.

Pricing and ordering

With the exception of our shared domain option, each of our paid TLS options require your agreement to purchase it. For specific details about pricing for each of Fastly's TLS options, see our pricing page (https://www.fastly.com/pricing) or contact sales@fastly.com (mailto:sales@fastly.com) for our current rates.

To order TLS service, contact customer support via email at support@fastly.com (mailto:support@fastly.com). Remember to put "TLS Certificate Request" in the subject of the request, and be sure to include your customer ID (/guides/account-management-and-

security/finding-and-managing-your-account-info#finding-your-customer-id) when you let us know which service option you're interested in. We'll walk you through the setup process for the option you choose.

Shared domain

This free option uses the Fastly SAN certificate's wildcard entry for `global.ssl.fastly.net`. To use this option, add a new domain in the Fastly web interface and set up an origin server for that domain. You can learn more about how to do that in our guide on setting up free TLS (/guides/securing-communications/setting-up-free-tls).

> ❗ **IMPORTANT:** If you DNS alias your own domain (`www.example.org`) to the shared domain (`example.global.ssl.fastly.net`), a TLS name mismatch warning will appear in the browser. The only way to fix the mismatch is by ordering one of the paid TLS options.

Shared TLS Certificate Service

This option uses the Fastly Subject Alternative Name (SAN) certificate. Specifically:

- You get to use your domain, but Fastly does the certificate administration.
- You provide your domain name list to Fastly and we add those names to the Certificate SAN field.

Our registrar explains the shared (SAN) certificate (https://www.globalsign.com/en/ssl/multi-domain-ssl/) as "a way to conserve IP addresses by putting multiple hostnames or domains on one certificate. There are no security implications….Addition of your name to the certificate still needs to be authorized by you."

Shared TLS Wildcard Certificate Service

This option uses the Fastly SAN certificate. Specifically:

- You get to use your domain, but Fastly does the certificate administration.
- You provide one or more wildcard domain name entries to Fastly and we add those names to the Certificate SAN field.

Hosting Service for Customer-Provided TLS Certificate

This hosting service option uses your TLS certificates, including Extended Verification (EV) certificates. Specifically:

- You provide the certificates to Fastly.

- Fastly installs those certificates into the caches and allocates IP addresses on each cache.
- Fastly creates a new (customer-specific) DNS Global Domain Map that associates the certificate with the allocated IP addresses.
- Fastly maintains the domain map and the certificate on the caches.

IMPORTANT: Fastly supports SHA-256 certificates signed by publicly trusted certificate authorities that have a minimum key size of 2048 bits for RSA. For performance reasons, we strongly recommend using a 2048-bit key size for RSA when larger key sizes are not required for your application.

Customer-Provided TLS Certificate Hosting Service

Fastly provides, on a limited availability basis, customer certificate hosting that uses Server Name Indication (SNI) for certificate selection. To see if your company meets the qualification criteria for this option, contact sales@fastly.com (mailto:sales@fastly.com). For this hosting option:

- You provide the certificates to Fastly.
- Fastly hosts your certificates on a shared domain map and uses SNI to select the correct certificate to serve based on the domain name sent.

IMPORTANT: All modern browsers support SNI. Clients that do not support SNI (such as those on Windows XP and Android 2.x or earlier) will see a certificate error.

NOTE: Any certificates provided by GlobalSign are subject to the terms of GlobalSign's Subscriber Agreement, which can be found at https://www.globalsign.com/repository/ (https://www.globalsign.com/en/repository/).

• Guides (/guides/) > Products and services > Fastly product lifecycle (/guides/fastly-product-lifecycle/)

Copyright 2018 Fastly, Inc.