

DGX Spark User Guide

NVIDIA Corporation

Nov 04, 2026

Release Notes

1 Overview **3**

2 Getting Started **5**
2.1 DGX Spark Release Notes . . . 5
2.1.1 Current Software Versions . . . 5
2.1.1.1 October 2026 Release . . . 6
2.1.1.2 July 2026 Release . . . 7
2.1.1.3 June 2026 Release . . . 7
2.1.1.4 April 2026 Release . . . 8
2.1.1.5 March 2026 Release . . . 8
2.1.1.6 February 2026 Release . . . 8
2.1.1.7 January 2026 Release . . . 9
2.1.1.8 November 2025 Release . . . 9
2.1.2 Known Issues . . . 9
2.2 Known Issues . . . 10
2.2.1 Use the supplied power adapter for optimal performance . . . 10
2.2.2 `nvidia-smi` reports “Memory-Usage: Not Supported” . . . 10
2.2.3 CUDA Support on DGX Spark . . . 10
2.2.4 Guidance for reporting memory resources with unified memory architecture . . . 10
2.2.5 Attached HDMI display goes into deep sleep mode after an extended period of inactivity and does not wake . . . 12
2.3 Hardware Overview . . . 12
2.3.1 System Overview . . . 12
2.3.1.1 Component Descriptions . . . 12
2.3.2 Physical Specifications . . . 13
2.3.2.1 Form Factor . . . 13
2.3.2.2 Environmental Requirements . . . 13
2.3.3 Connectivity and I/O . . . 13
2.3.3.1 Rear Panel . . . 13
2.3.4 Performance Specifications . . . 14
2.3.4.1 Compute Performance . . . 14
2.3.4.2 AI/ML Capabilities . . . 14
2.3.5 Power and Thermal Management . . . 14
2.3.5.1 Power Requirements . . . 14
2.3.5.2 Thermal Management . . . 15
2.4 Initial Setup - First Boot . . . 15
2.4.1 What You’ll Do . . . 15
2.4.2 Choose how to access your system during initial setup . . . 15
2.4.3 Get Ready . . . 16
2.4.4 Run the First-Time Setup . . . 16
2.4.4.1 Getting Started . . . 17
2.4.4.2 What to Expect During Setup . . . 17
2.4.5 Next Steps . . . 19

2.5 System Configuration and Operation . . . 20
2.5.1 System Overview . . . 20
2.5.1.1 Flexible Access and Usage . . . 20
2.5.1.2 Key Capabilities . . . 20
2.5.1.3 System Architecture . . . 20
2.5.1.4 Software . . . 21
2.5.1.5 Getting Started . . . 21
2.5.2 UEFI Settings . . . 21
2.5.2.1 Access UEFI . . . 21
2.5.2.2 UEFI Documentation . . . 22
2.5.2.3 Enable or Disable WiFi and Bluetooth . . . 22
2.5.2.4 Change Display Reserved Memory . . . 22
2.5.2.5 Boot from a USB Device . . . 23
2.5.2.6 Enable or Disable Secure Boot . . . 23
2.5.2.7 Configure PXE Boot . . . 23
2.5.2.8 Troubleshooting . . . 24
2.5.3 ConnectX-7 Networking . . . 24
2.5.3.1 Connecting DGX Spark Systems into a Cluster . . . 24
2.6 Software . . . 28
2.6.1 DGX Dashboard . . . 29
2.6.1.1 Overview . . . 29
2.6.1.2 Integrated JupyterLab . . . 29
2.6.1.3 Accessing the Dashboard . . . 30
2.6.1.4 Administrator Policy Settings . . . 30
2.6.2 NVIDIA Sync . . . 31
2.6.2.1 Overview . . . 31
2.6.2.2 Key Capabilities . . . 31
2.6.2.3 Using NVIDIA Sync with Your DGX Spark . . . 31
2.6.2.4 DGX Spark Skills . . . 32
2.6.3 DGX OS . . . 32
2.6.3.1 Overview . . . 32
2.6.3.2 Security and Compliance . . . 32
2.6.3.3 Release Cadence . . . 32
2.6.4 NVIDIA Nsight . . . 33
2.6.4.1 Overview . . . 33
2.6.4.2 Latest Versions . . . 33
2.6.5 NVIDIA Container Runtime for Docker . . . 33
2.6.5.1 Overview . . . 33
2.6.5.2 Installation . . . 33
2.6.5.3 Usage . . . 34
2.6.5.4 Validation . . . 34
2.6.5.5 Troubleshooting . . . 35
2.6.6 NGC . . . 36
2.6.6.1 Overview . . . 36
2.6.6.2 Access NVIDIA NIM Containers . . . 36
2.6.6.3 Getting Started . . . 37
2.6.6.4 Basic Usage . . . 37
2.6.6.5 Common Workflows . . . 38
2.6.6.6 Best Practices . . . 38
2.6.6.7 Troubleshooting . . . 39
2.6.7 NVIDIA AI Enterprise—DGX Spark Quick Start Guide . . . 39
2.6.7.1 Set Up Your DGX Spark System . . . 40
2.6.7.2 NIM Access and NVIDIA AI Enterprise Entitlement . . . 40
2.6.7.3 Optional: NVIDIA AI Enterprise—DGX Spark Evaluation . . . 40

2.6.7.4 Order Confirmation Message . . . 40
2.6.7.5 Create Your NVIDIA AI Enterprise Account . . . 40
2.6.7.6 Access NVIDIA AI Enterprise—DGX Spark Software Asset . . . 41
2.7 Common Use Cases . . . 41
2.8 PXE Boot Setup . . . 41
2.8.1 Requirements . . . 41
2.8.2 Overview of the PXE Server . . . 42
2.8.2.1 PXE Server Configuration for DGX OS Image . . . 42
2.8.2.2 PXE Server Configuration for Recovery Media . . . 44
2.8.3 TFTP and HTTP Server Verification . . . 45
2.8.4 Useful Parameters for Configuring Your System's Network Interfaces . . . 46
2.8.5 Parameters Unique to the Spark OS Installer . . . 46
2.8.6 Configure Your DHCP Server . . . 46
2.8.7 Enable PXE Boot on the DGX Spark . . . 47
2.9 Enterprise Manageability . . . 49
2.9.1 Enterprise Lifecycle Integration . . . 49
2.9.1.1 How to Use This Information . . . 49
2.9.1.2 Support Boundaries and Operational Responsibilities . . . 51
2.9.1.3 DGX Spark Overview, Enterprise Challenges, Lifecycle Backbone, and the JSON/Agentless SSH . . . 53
2.9.1.4 Integration Patterns: SSH Execution from Management Platforms . . . 62
2.9.1.5 Implementation . . . 67
2.9.1.6 Clear Asset Information (Seven Tools) . . . 74
2.9.1.7 Controlled Software/Firmware Updates (One Tool) . . . 74
2.9.1.8 Remote Ops and Remediation (Two Tools) . . . 74
2.9.1.9 Tool Locations and Reference Code Map . . . 77
2.9.1.10 Platform Wrapper Patterns and Packaging Examples . . . 81
2.9.2 Custom Installation with cloud-init . . . 87
2.9.2.1 Overview . . . 87
2.9.2.2 Cloud-init Support . . . 87
2.9.2.3 IT Administrator Workflows . . . 87
2.9.2.4 Prerequisites and Source Media . . . 88
2.9.2.5 USB and Image Layout . . . 89
2.9.2.6 First Boot Behavior . . . 90
2.9.2.7 Where to Place Customization Content . . . 90
2.9.2.8 OEMDATA URL and List File Reference . . . 91
2.9.2.9 Air-Gapped and Custom Installation Patterns . . . 92
2.9.2.10 Example Constants . . . 94
2.9.2.11 USB Partitioning and the OEMDATA Layout . . . 94
2.9.2.12 Host a Minimal APT Repository and Firmware Tree . . . 96
2.9.2.13 Mirror the Full Ubuntu Ports and LVFS Content on a Server . . . 100
2.9.2.14 Security Considerations . . . 105
2.9.2.15 Verify the Customization and Installation Outcomes . . . 106
2.9.2.16 Prepare the Installation Media and Client (Verification Flow) . . . 107
2.9.2.17 Reference: Scripts and cloud-init . . . 107
2.9.2.18 Validation Scenarios and Feedback Questions . . . 127
2.10 OS and Component Update Guide . . . 129
2.10.1 Ubuntu Support . . . 129
2.10.2 Update Methods . . . 129
2.10.3 Using DGX Dashboard for Updates . . . 129
2.10.4 Manual System Updates . . . 130
2.10.5 Update Best Practices . . . 130
2.11 System Repair . . . 131
2.11.1 Previously Installed Kernel . . . 131

2.11.1.1 Boot from the GRUB Menu . . . 131
2.11.1.2 Common Scenarios for the Booted Kernel . . . 133
2.11.2 Bootable USB . . . 135
2.11.2.1 Download the Ubuntu Desktop ISO . . . 135
2.11.2.2 Boot into Ubuntu . . . 135
2.11.2.3 Common Scenarios for the Live USB . . . 137
2.12 System Recovery . . . 140
2.12.1 System Recovery Overview . . . 141
2.12.1.1 Recovery Requirements . . . 141
2.12.1.2 Recovery Process Steps . . . 141
2.13 Get the Right Support for Your DGX Spark . . . 145
2.13.1 Passthrough Virtualization Support . . . 145
2.13.2 NVIDIA DGX Spark Hardware Support . . . 145
2.13.3 NVIDIA AI Enterprise—DGX Spark Support . . . 146
2.13.4 Enterprise Manageability . . . 146
2.13.5 Security and Vulnerability Response . . . 146
2.13.6 Field Diagnostic Software . . . 146
2.13.6.1 Removing a Previous Version . . . 146
2.13.6.2 Package Dependencies . . . 146
2.13.6.3 ConnectX-7 Test Prerequisites . . . 147
2.13.6.4 Installing the Field Diagnostic Software . . . 147
2.13.6.5 Verifying Tool Installation . . . 147
2.13.6.6 Running Field Diagnostics . . . 147
2.14 Regulatory Compliance Information . . . 148
2.14.1 Safety Information . . . 148
2.14.1.1 Anti-Static Precautions . . . 149
2.14.1.2 Symbols on Equipment . . . 149
2.14.1.3 General Precautions . . . 150
2.14.1.4 Power Rating . . . 151
2.14.1.5 USB-C Ports Rating . . . 151
2.14.2 Compliance . . . 151
2.14.2.1 United States . . . 151
2.14.2.2 Canada . . . 153
2.14.2.3 European Union . . . 154
2.14.2.4 Great Britain (England, Wales and Scotland) . . . 156
2.14.2.5 Australia and New Zealand . . . 157
2.14.2.6 Japan . . . 157
2.14.2.7 China . . . 159
2.14.2.8 South Korea . . . 163
2.14.2.9 Taiwan . . . 166
2.14.2.10 India . . . 168
2.14.2.11 Singapore . . . 169
2.14.2.12 Malaysia . . . 169
2.14.2.13 Bluetooth . . . 169
2.15 IMPORTANT NOTICE - PLEASE READ AND AGREE BEFORE USING NVIDIA SOFTWARE OFFERINGS . . . 169
2.16 Third-Party License Notices . . . 187
2.16.1 LIBICONV Library . . . 187
2.17 Notices . . . 200
2.17.1 Notice . . . 200
2.17.2 Trademarks . . . 201

This guide is also available for download as a PDF.

Chapter 1. Overview

The DGX Spark is NVIDIA's compact AI computer designed for developers, data scientists, and AI researchers who need powerful computing capabilities for AI development and deployment.

Chapter 2. Getting Started

2.1. DGX Spark Release Notes

This section provides release notes for the DGX Spark, including information about new features, known issues, and software version updates.

2.1.1. Current Software Versions

The following table shows the current version information for the DGX Spark software stack.

Component	Version
NVIDIA DGX OS	7.6.0
NVIDIA GPU Driver	580.178.04
NVIDIA CUDA Toolkit	13.0.3
NVIDIA TensorRT	10.14.1
Canonical Kernel	7.0.0-1019
UEFI	1.111.17
Embedded Controller (EC)	3.13.7
USB Power Delivery (USB PD)	0.5.25
Trusted Platform Module (TPM)	7.516.1
System on Chip (SoC)	2.158.15

Note

These release versions apply only to the DGX Spark Founders Edition. GB10-based partner systems may not receive updates at the same time.

2.1.1.1 October 2026 Release

2.1.1.1.1 What's New

- **DGX Dashboard Wi-Fi Assistant**: The new feature enables users to join a new Wi-Fi network when DGX Spark is used in appliance mode (without a keyboard, mouse, and display).
- **Dynamic Display Reserved Memory**: Display Reserved Memory adds a UEFI option to automatically optimize the memory carveout to match display usage. Systems without connected displays can automatically adjust to the minimum carveout (512 MB).
- **DGX Spark Skills (available in late October)**: Workflow-focused resources for agents will be available through NVIDIA Sync (OpenCode). These resources provide DGX Spark-specific knowledge and guidance for troubleshooting, clustering multiple DGX Spark systems, device maintenance, playbook discovery, support escalation, awareness of official documentation, convenient user forum search, and other DGX Spark workflows. Refer to *DGX Spark Skills* for details on how to implement the skills for existing agent harnesses.
- **New Kernel Support**: This update supports the Linux 7.0 kernel (previously 6.17). It includes security updates and improved file and memory management:
  - Large page frame number maps (pfnmap)
  - Splitting extended GPU memory (EGM) into 1 GB chunks
  - Memory System Resource Partitioning and Monitoring
- **Support for 64 GB Systems**: The system software now supports both 64 GB and 128 GB configurations, so users do not have to match software updates with hardware. Dashboards and other system information automatically reflect the available memory.
- **Native Cloud-init Support**: Simple configuration files can automate system configuration and provisioning at first boot through the *Cloud-init* initialization framework. Enterprise environments can immediately deploy custom configurations at first boot.

2.1.1.1.2 Fixed Issues

- **DKMS Kernel-Module Build Failures**: Addressed DKMS kernel-module build failures after an `nvidia-installer` installation.
- **Delayed Wake-Up**: Addressed delayed wake-up when multiple threads wait on the same Vulkan semaphore.
- **Black Screens After Modesets**: Addressed black screens after modesets in X11 applications using the Present extension.
- **OpenGL Buffers Allocated With glBufferStorage**: Addressed an issue where OpenGL buffers allocated with `glBufferStorage` and no storage flags migrated from GPU memory to host memory.
- **Fans Stop Responding in Headless Mode**: Addressed an issue where the fans stopped responding when the system was in headless (appliance) mode at low idle power draw.
- **Ukrainian Language Option**: Addressed an issue where the initial system configuration screens did not offer the Ukrainian language option. The option was available only after login in *Settings*.

2.1.1.2 July 2026 Release

2.1.1.2.1 What's New

- **Improved Memory Management**: The included driver enhances Out-of-Memory (OOM) handling with GB10's unified memory architecture. User feedback is now available when the system encounters memory pressure, which improves robustness and reliability when operating larger models.
- **Adjustable Display Reserved Memory**: The Display Reserved Memory can now be toggled between 2 GB (default) and 4 GB through the system BIOS. Workflows that require displaying many applications and windows will benefit from the larger display memory carveout. Users in appliance mode do not need to change default settings.
- **Enterprise Management Feature Enhancements**: The cloud-init initialization framework and image customization include the following updates. Refer to Cloud-init for DGX Spark for more information.
  - **Cloud-init Network Protocol Support**: In addition to direct USB connections, Cloud-init now supports DHCP, HTTP, and TFTP.
  - **FastOS Recovery Image Customization**: In addition to already supporting the customization of DGX OS, Cloud-init adds support to customize the NVIDIA-provided FastOS recovery image for the DGX Spark Founders Edition.

2.1.1.2.2 Fixed Issues

- **System and Display Instability When Hot-Plugging Displays**: Addressed system and display instability when hot-plugging displays.

2.1.1.3 June 2026 Release

2.1.1.3.1 What's New

- **Out-of-Box-Experience (OOBE) Enhancements**: Updates to the OOBE that enable faster initial device setup time and easier navigation to discover and install local AI agents, including:
  - **Over-the-Air (OTA) Updates**: OTA updates are not installed by default during initial setup, allowing users to start using their system sooner. Users can download and install OTA updates after initial system setup.
  - **NemoClaw Playbook**: After initial system boot, the DGX Spark playbook site opens, where the NemoClaw playbook is prominently displayed for streamlined setup of a sandboxed local agent.
- **New Application and Library Features**: NVIDIA Sync and NVIDIA Collective Communications Library (NCCL) include the following updates:
  - **NVIDIA Sync Cluster Assistant**: The Cluster Assistant is now available on the **Settings** page in the NVIDIA Sync application. The Cluster Assistant helps users connect up to three devices without a network switch, and up to four devices using a switch.
  - **NCCL**: NCCL (Version 2.30u1) has updated support for connecting three DGX Spark systems in a ring topology.

2.1.1.4 April 2026 Release

2.1.1.4.1 What's New

- **Enterprise Features**: Support for tools and capabilities designed to facilitate system management in enterprise environments, including:
  - **Enterprise Management Guide**: Enables IT administrators to plan, configure, and manage DGX Spark deployments at scale with a dedicated reference covering workflows, provisioning, update procedures, and system lifecycle management across fleets. For more information, refer to *Enterprise Manageability*.
  - **Skip the Automated OOBE for IT Administrator Provisioning**: Allows IT administrators to bypass the OOBE entirely during provisioning, reducing manual per-device setup steps and minimizing touch while deploying across multiple DGX Spark units.
  - **USB and Local Repository Support for Installations and Updates**: Allows IT administrators to deliver OS installations and software updates through USB drives or internal local package repositories, removing the dependency on cloud connectivity for software distribution and patch management, and allowing administrators to control when new releases are installed.
  - **Support for Air Gapped Deployment and Updates**: Enables IT administrators to deploy and operate DGX Spark systems on isolated networks with no external internet access, suitable for strict security, compliance, and data sovereignty requirements.
  - **Customized Enterprise ISOs Through cloud-init**: Allows IT administrators to embed site-specific configurations (users, network settings, software, and policies) directly into custom DGX OS images using cloud-init, ensuring every unit arrives pre-configured to enterprise standards on first boot.
- **DGX Dashboard Enhancements**: Release highlights are now provided for software releases, enabling users to determine the urgency of applying updates.

2.1.1.4.2 Fixed Issues

- **Bluetooth Keyboard Pairing**: Addressed an issue where the Bluetooth keyboard pairing modal did not appear in some instances when a keyboard was plugged in.
- **Unable to Detect Network During System Setup**: Addressed Wi-Fi and Ethernet connectivity issues reported during system setup.

2.1.1.5 March 2026 Release

2.1.1.5.1 What's New

- **Partner Factory Updates**: Firmware update to facilitate GB10 partner factory updates.

2.1.1.6 February 2026 Release

2.1.1.6.1 Fixed Issues

- **Improved Performance with Multiple DGX Spark Systems**: Addressed performance regression noted by some users with multiple connected DGX Spark systems after updating to DGX OS 7.4.0.

2.1.1.7 January 2026 Release

2.1.1.7.1 What's New

- **Power Management**: Added hot plug support for the ConnectX-7 network adapter, saving up to 18W of power when the adapter is not in use.
- **Bluetooth Audio Support**: Support for Bluetooth audio devices, including headphones and headsets, is now available.
- **Additional Security Controls**: Wi-Fi and Bluetooth can now be disabled in the Unified Extensible Firmware Interface (UEFI) for environments requiring additional security.

2.1.1.7.2 Fixed Issues

- **Improved Monitor Support**: Better compatibility with TVs and monitors, including multi-monitor configurations and the use of non-native resolutions and refresh rates.
- **Improved Peripheral Setup**: Smoother and more complete OOBE navigation.

2.1.1.8 November 2025 Release

2.1.1.8.1 What's New

- **New DGX OS kernel**: The operating system now incorporates the Ubuntu 6.14 Hardware Enablement (HWE) kernel stack. The new kernel brings ongoing performance gains, enhanced stability, broader hardware compatibility, and the latest security updates for a more secure operating environment.
- **JupyterLab updated to latest CUDA and PyTorch**: Updated JupyterLab to CUDA 13.0.2 and the latest PyTorch version, enabling users to work with updated frameworks immediately without extra downloads.

2.1.1.8.2 Fixed Issues

- **Improved memory reporting in DGX Dashboard**: Addressed the issue with memory reporting differences with unified memory architecture. The readout is now consistent with CUDA guidance for unified memory systems. For more information, refer to CUDA unified memory guidance.
- **Image Generation in JupyterLab**: Resolved the inability to generate images in Stable Diffusion XL Playbook example. Users can now run end-to-end example workflow inside JupyterLab.
- **Improved Peripheral Interoperability**: Better compatibility with USB-C devices, monitors, Bluetooth peripherals, and Wi-Fi access points.
- **Improved recovery image reliability**: The recovery image now installs correctly on macOS and also when multiple external USB-C drives are connected.
- **Keyboard accessibility improvements**: Smoother and more complete OOBE navigation, including the ability to complete entire system setup process using only keyboard.

2.1.2. Known Issues

For an updated list of known issues, refer to *Known Issues*.

2.2. Known Issues

This topic provides a summary of known issues with DGX Spark systems.

2.2.1. Use the supplied power adapter for optimal performance

For optimal performance, use the supplied power adapter with the DGX Spark system. Using a different adapter may reduce performance, prevent boot, or cause unexpected shutdowns.

2.2.2. `nvidia-smi` reports "Memory-Usage: Not Supported"

On iGPU platforms, `nvidia-smi` will display "Memory-Usage: Not Supported" even though per-process GPU memory is listed. This is expected because iGPUs do not have dedicated framebuffer memory.

2.2.3. CUDA Support on DGX Spark

The CUDA version on your DGX Spark device has been verified to work with your system hardware at the time of software update release. The latest features and performance improvements are available through NVIDIA NGC containers (for example, PyTorch, vLLM, and TensorRT-LLM).

2.2.4. Guidance for reporting memory resources with unified memory architecture

NVIDIA is actively working with third-party ecosystem partners to bring their software to DGX Spark. For example, we have provided direction on implementing memory management on systems based on a unified memory architecture (UMA) to help ensure accurate reporting of available resources.

DGX Spark systems use a unified memory architecture (UMA), where the GPU shares system memory (DRAM) with the CPU and other compute engines. This design reduces latency and allows larger amounts of memory to be used for GPU workloads. On UMA systems, the CPU can dynamically manage DRAM contents, including freeing up memory by swapping pages between DRAM and the system's SWAP area. However, the `cudaMemGetInfo` API does not account for memory that could potentially be reclaimed from SWAP. As a result, the memory size reported by `cudaMemGetInfo` may be smaller than the actual allocatable memory, since the CPU may be able to release additional DRAM pages by moving them to SWAP.

To more accurately estimate the amount of allocatable device memory on DGX Spark platforms, CUDA application developers should consider the possibility of DRAM reclamation via SWAP and not rely solely on the values returned by *cudaMemGetInfo*. The following provides an example implementation using C standard libraries:

```
#include <stdio.h>

int getAvailableMemory(long *availableMemoryKb, long *freeSwapKb) {
```

(continues on next page)

(continued from previous page)

```
    FILE *meminfoFile = NULL;
    char lineBuffer[256];
    long hugeTlbTotalPages = -1;
    long hugeTlbFreePages = -1;
    long hugeTlbPageSize = -1;

    if (availableMemoryKb == NULL || freeSwapKb == NULL) {
        return 1;
    }

    meminfoFile = fopen("/proc/meminfo", "r");
    if (meminfoFile == NULL) {
        return 1;
    }

    *availableMemoryKb = -1;
    *freeSwapKb = -1;

    while (fgets(lineBuffer, sizeof(lineBuffer), meminfoFile)) {
        long value;

        if (sscanf(lineBuffer, "MemAvailable: %ld kB", &value) == 1) {
            *availableMemoryKb = value;
        } else if (sscanf(lineBuffer, "SwapFree: %ld kB", &value) == 1) {
            *freeSwapKb = value;
        } else if (sscanf(lineBuffer, "HugePages_Total: %ld", &value) == 1) {
            hugeTlbTotalPages = value;
        } else if (sscanf(lineBuffer, "HugePages_Free: %ld", &value) == 1) {
            hugeTlbFreePages = value;
        } else if (sscanf(lineBuffer, "Hugepagesize: %ld kB", &value) == 1) {
            hugeTlbPageSize = value;
        }

        if (*availableMemoryKb != -1 &&
            *freeSwapKb != -1 &&
            hugeTlbTotalPages != -1 &&
            hugeTlbFreePages != -1 &&
            hugeTlbPageSize != -1) {
            break;
        }
    }

    fclose(meminfoFile);

    if (hugeTlbTotalPages != 0 && hugeTlbTotalPages != -1) {
        *availableMemoryKb = hugeTlbFreePages * hugeTlbPageSize;

        // Hugetlbfs pages are not swappable.
        *freeSwapKb = 0;
    }

    return 0;
```

(continues on next page)

(continued from previous page)

```
}
```

As a workaround for debugging purposes, you can flush the buffer cache manually with the following command:

```
sudo sh -c 'sync; echo 3 > /proc/sys/vm/drop_caches'
```

After flushing the cache, restart your application.

2.2.5. Attached HDMI display goes into deep sleep mode after an extended period of inactivity and does not wake

In some cases, after a long period of inactivity, the attached HDMI display might go into a deep sleep mode and not wake up when you press a key or move the mouse. If this happens, press a physical button on the monitor to wake it up.

2.3. Hardware Overview

Powered by the NVIDIA Grace Blackwell architecture, DGX Spark enables developers, researchers, and data scientists to prototype, deploy, and fine-tune large AI models on their desktop. This section provides information about the hardware components and specifications.

2.3.1. System Overview

The DGX Spark features:

- NVIDIA Grace Blackwell architecture with integrated GPU and CPU
- 20-core Arm processor with high-performance cores
- 128 GB unified system memory
- Compact desktop form factor
- Advanced connectivity including Wi-Fi 7, 10 GbE, and ConnectX-7
- Support for AI models up to 200 billion parameters (or 405B for dual-Spark configuration)

2.3.1.1 Component Descriptions

The DGX Spark includes the following components:

Table 1: Component Specifications

Component	Specification
GPU	NVIDIA Blackwell Architecture with 5th Generation Tensor Cores, 4th Generation RT Cores
CPU	20-core Arm processor (10 Cortex-X925 + 10 Cortex-A725)
Memory	128 GB LPDDR5x unified system memory, 256-bit interface, 4266 MHz, 273 GB/s bandwidth
Storage	1 TB or 4 TB NVMe M.2 with self-encryption
Network	1x RJ-45 (10 GbE), ConnectX-7 Smart NIC, Wi-Fi 7, Bluetooth 5.4
Connectivity	4x USB Type-C, 1x HDMI 2.1a, HDMI multichannel audio
Video Processing	1x NVENC, 1x NVDEC

2.3.2. Physical Specifications

2.3.2.1 Form Factor

- **Chassis Type**: Small form factor (SFF)
- **Dimensions**: 150 mm (L) x 150 mm (W) x 50.5 mm (H)
- **Weight**: 1.2 kg (2.6 lbs)

2.3.2.2 Environmental Requirements

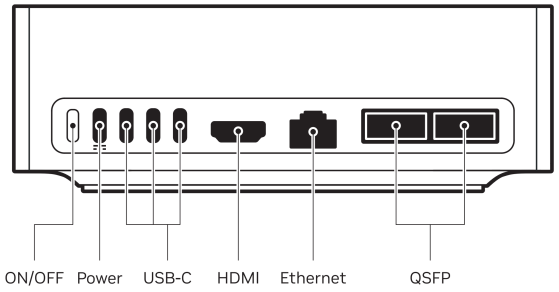
Table 2: Environmental Specifications

Specification	Value
Ideal Operating Temperature	5°C to 30°C (41°F to 86°F)
Operating Humidity	10% to 90% (non-condensing)

2.3.3. Connectivity and I/O

2.3.3.1 Rear Panel

- Power button
- 4x USB Type-C (one for power delivery)
- 1x HDMI 2.1a display connector
- 1x RJ-45 Ethernet connector (10 GbE)
- 2x QSFP Network connectors (ConnectX-7)



2.3.4. Performance Specifications

2.3.4.1 Compute Performance

- **AI Compute**: Up to 1,000 TOPS (trillion operations per second) inference and up to 1 PFLOP (petaFLOP) at FP4 precision with sparsity
- **CUDA Cores**: 6,144
- **Copy Engines**: 2 (enables simultaneous data transfers to and from GPU memory, improving throughput for AI workloads)
- **CPU Performance**: 20 cores (10 Cortex-X925 + 10 Cortex-A725)
- **Memory Bandwidth**: 273 GB/s
- **Memory Channels**: 16 channels (256 bit) LPDDR5X 8533

2.3.4.2 AI/ML Capabilities

- **Model Support**: AI models up to 200 billion parameters
- **Tensor Performance**: 5th Generation Tensor Cores with FP4 support
- **Framework Support**: PyTorch, TRT-LLM, and other AI frameworks
- **Use Cases**: Inference, deployment, and fine-tuning of large language models

2.3.5. Power and Thermal Management

2.3.5.1 Power Requirements

- **Power Supply**: 240W external power supply (included)
  - GB10 SOC Thermal Design Power (TDP) is 140W
  - 100W is available for other system components (ConnectX-7, Wi-Fi, SSD, USB-C, etc.)
- **Usage Requirement**: Use of the provided 240W power supply is required for optimal performance. Using a different or lower-rated power supply may result in reduced system performance, failure to boot, or unexpected shutdowns.
- **Input Voltage**: Standard AC power input

2.3.5.2 Thermal Management

- **Cooling Solution**: Integrated thermal management system
- **Form Factor**: Compact design optimized for desktop placement
- **Ideal Operating Temperature**: 5°C to 30°C (41°F to 86°F)

2.4. Initial Setup - First Boot

This guide walks you through setting up your DGX Spark for the first time. You'll choose how to access your system during the initial setup, and run the first-time setup utility to configure everything. The access method you choose is only for completing the initial setup - after setup is complete, you can access your DGX Spark any way you like: locally with a monitor and keyboard, over the local network from another computer, or a mix of both.

Important

Use of the NVIDIA CUDA Toolkit preinstalled on DGX Spark is subject to the NVIDIA CUDA Toolkit End User License Agreement.

2.4.1. What You'll Do

This setup process includes:

- Choosing how to access the system during initial setup (with a display, or as a network appliance)
- Preparing your system and connections
- Running the first-time setup utility to configure your system

2.4.2. Choose how to access your system during initial setup

To complete the initial setup, you'll need to access your DGX Spark. You can do this in one of two ways:

With a Display (Local Setup)

- Connect keyboard and mouse via USB or Bluetooth
- Connect a display to work directly on the system
- Follow the setup wizard on screen

Over the Network (as a Network Appliance)

- Access the system over your local network from another computer
- Use another computer to complete setup via web browser
- No Spark display or keyboard required for the setup process

Note

This choice is only about how you'll complete the initial setup process. After setup is finished, you can access your DGX Spark however you prefer. You're not locked into your original choice.

2.4.3. Get Ready

Important

The DGX Spark device starts up immediately when power is applied. Please attach all peripherals (display, keyboard, mouse, network, etc.) before connecting the power supply.

Before starting, ensure you have:

- A fast, reliable internet connection (Wi-Fi or Ethernet) to download required updates during the initial setup process. Connections using captive portals (such as hotel or airport Wi-Fi) or those prone to disconnections (like phone hotspots) are not recommended. If you do not have access to a stable connection, consider downloading the system recovery media and using *System Recovery* to install the latest software for your DGX Spark.
- For Local Setup: A display, keyboard, and mouse connected (or available using Bluetooth).
- For Network Setup: A computer on the same network to access the setup interface.
- Power connected to the system (the system will start automatically when power is applied).

Note

Display Troubleshooting: Some displays may have trouble with Spark out of the box. If you are connecting over USB-C/DisplayPort and there is no display, try using HDMI instead.

Note

If you plan to use a wired network connection, plug in the network cable before starting the installation. This helps avoid connection issues later in the process.

2.4.4. Run the First-Time Setup

The first-time setup utility will guide you through:

- Powering on and initializing the system
- Selecting your preferred setup mode
- Downloading and installing critical updates
- Completing your initial configuration

Warning

Critical: Do not shut down or reboot the system during the update process. The installation cannot be interrupted once the download begins, and powering down during updates can cause system damage.

2.4.4.1 Getting Started

The way you start the installation depends on your chosen access method:

With a Display (Local Setup)

1. Power on the system
2. The first-time setup utility will start automatically on the connected display
3. Use your wired keyboard and mouse (already connected) to navigate
4. If a keyboard or mouse is not detected, you will be prompted to put your Bluetooth devices in pairing mode:

   USB devices can be plugged in at any time and should start working, even if detected improperly. Bluetooth devices can be put into pairing mode and will generally still pair while on the "Get Started" screen (exception - keyboards that require a passcode to type in won't work on this screen). Once you click on "Get Started," Bluetooth pairing stops, so you will have to power cycle to try again.
5. Follow the on-screen prompts to complete the setup process

Over the Network (as a Network Appliance)

1. Power on the system. This creates a Wi-Fi hotspot that you will use to connect to the system and continue the setup process. The SSID and password for the Wi-Fi hotspot are printed on a sticker attached to the Quick Start Guide included with your DGX Spark's packaging
2. From another computer, connect to the Spark's Wi-Fi hotspot using the SSID and password provided on the Quick Start Guide. A captive portal page will open in the default web browser on your computer. If it does not open automatically, use your browser to navigate to the Spark's system setup page listed on the Quick Start Guide.
3. Follow the on-screen prompts to continue the setup process.

   When the DGX Spark joins your home network, its Wi-Fi hotspot will turn off and your computer will reconnect to the device through your Wi-Fi network to resume the setup process. If you are not able to connect to the DGX Spark after it joins your home network, you must connect a display/keyboard/mouse to continue.

   Note

   It may take as long as 10 minutes for the DGX Spark to install all of it's updates and join your home network.

2.4.4.2 What to Expect During Setup

The first-time setup utility will guide you through several configuration steps. Simply follow the on-screen prompts to complete each step.

Setup Process Steps:

1. **Language and Time Zone Selection**

   Choose your preferred language and time zone settings for the system. Note that the input fields will filter as you type.

2. **Keyboard Layout Selection** (Local Setup only)

   Select your keyboard layout (e.g. US keyboard vs. Russian keyboard). This screen only appears when using a display during setup.

3. **Terms and Conditions**

   Review and accept the terms and conditions to continue with the installation.

4. **User Account Creation**

   Create your username and password for system access.

5. **Information Sharing Settings** (Optional)

   Configure analytics and crash reporting preferences. You can skip this step if desired.

6. **Wi-Fi Network Selection**

   Select your Wi-Fi network. This step is automatically skipped if an Ethernet cable that is providing internet access is connected.

7. **Wi-Fi Password**

   Enter the password for your selected Wi-Fi network.

8. **Joining Wi-Fi Network**

   The system connects to your Wi-Fi network and tears down the access point. Your computer will automatically reconnect to your default network.

   Note

   Network Connection Issues:

   - If your computer automatically reconnects to the same network as the Spark, the installation should continue seamlessly
   - If not, you'll need to connect your computer to the same network as the Spark while the setup app is waiting for the network setup process to complete
   - If the setup fails, you must connect a display/keyboard/mouse to continue
   - The modal instructs you to try to reconnect to the Spark's hotspot and try again. This will work if the Spark actually failed to join the network (e.g. wrong password) versus your laptop can't communicate with the Spark
   - If you DO NOT see the hotspot available when this error modal appears, that means the Spark did join the network but your laptop can't communicate with it. This could be because:
     - Device isolation
     - You failed to join the same network as your Spark
     - mDNS does not work on your network due to its configuration (e.g. a complex corporate network)

9. **Software Download and Installation**

After connecting to the network, the system will automatically download and install the full software image. This process can take some time, and the system may reboot more than once before it's complete. If you are setting up over the network from another computer, you will not be able to access the device during this time. The process may continue for up to 10 minutes after the interface shows that the device is rebooting to finish the setup

> **Warning**
>
> **Do not shut down or reboot the system during this process.** The installation cannot be interrupted once the download begins.

10. **Installation Complete**

    The device will reboot automatically when installation is complete, and you can then use it normally.

2.4.5. Next Steps

Congratulations! Your DGX Spark is now ready to use. Here are the recommended ways to access and start working with your system:

Access Your System

- **Locally**: Use your DGX Spark like any desktop computer with a monitor, keyboard, and mouse
- **Over the Network**: Connect to your DGX Spark from another computer on the same network using the NVIDIA Sync tool (see *NVIDIA Sync*), SSH, or remote desktop
- **Mix Both**: Use whatever access method fits your current workflow - you can switch freely between local and network access
- **Dashboard Access**: Use the built-in DGX Dashboard for system monitoring, updates, and JupyterLab access. See *DGX Dashboard* for more information

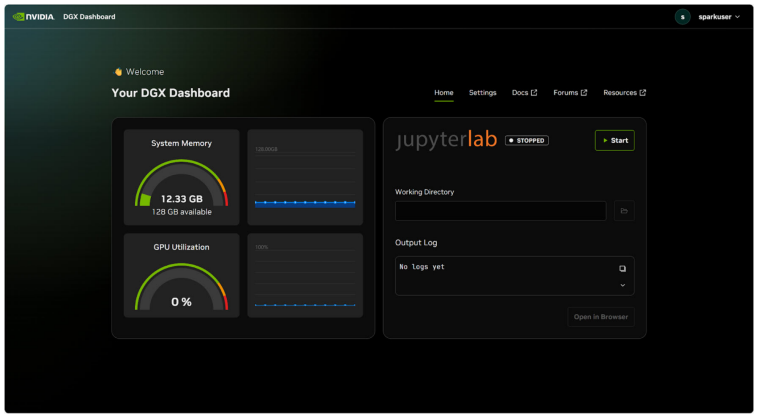


Fig. 1: The DGX Dashboard provides an intuitive interface for system monitoring and development

Additional Resources

- Visit the NVIDIA Spark Developer Portal at https://build.nvidia.com/spark for the latest guides, tutorials, and updates

- Refer to *DGX Spark Release Notes* for the latest software updates and features
- See *Known Issues* for troubleshooting common problems

Your DGX Spark is now ready to power your AI development and deployment workflows!

2.5. System Configuration and Operation

Configuring and operating your DGX Spark effectively is key to delivering consistent results across AI/ML workflows. This section provides an overview of the platform, recommended UEFI settings, clustering procedures, and foundational security guidance to help you deploy, manage, and scale with confidence.

2.5.1. System Overview

Powered by the NVIDIA Grace Blackwell architecture, DGX Spark enables developers, researchers, and data scientists to prototype, deploy, and fine-tune large AI models on their desktop.

2.5.1.1 Flexible Access and Usage

The DGX Spark is designed for maximum flexibility in how you access and use it. You can seamlessly switch between different access methods based on your needs:

- **Local Access**: Connect a keyboard, mouse, and monitor to work directly on the system
- **Network Access**: Access your system from another computer on the same network using SSH, *NVIDIA Sync*, or remote desktop tools
- **Hybrid Usage**: Mix and match access methods - work locally one day and over the network the next, or even simultaneously

All access methods are fully supported and equally capable. Your DGX Spark adapts to your workflow, whether you're working at your desk with a monitor or accessing it remotely as a network appliance on the same network.

2.5.1.2 Key Capabilities

Your DGX Spark enables you to:

- **Run Inference**: Deploy models for real-time AI applications
- **Develop AI Models**: Train and fine-tune models with up to 200 billion parameters
- **Process Data**: Handle large datasets with high-performance computing
- **Experiment Freely**: Test new ideas without cloud computing costs
- **Scale Workloads**: Connect multiple systems for larger projects

2.5.1.3 System Architecture

The DGX Spark is built on NVIDIA's Grace Blackwell architecture, providing:

- **Unified Memory**: 128 GB of high-bandwidth memory for large models
- **High-Performance Computing**: 20-core ARM64-based processor with integrated GPU
- **Advanced Connectivity**: Wi-Fi 7, 10 GbE, ConnectX-7 NIC, and multiple I/O options

- **Compact Form Factor**: 150mm x 150mm x 50.5mm desktop design

For detailed hardware specifications, see *Hardware Overview*.

2.5.1.4 Software

Your system comes pre-configured with:

- **NVIDIA DGX OS**: Optimized operating system for AI workloads
- **Development Tools**: CUDA, cuDNN, and NVIDIA's development ecosystem
- **Container Support**: Docker and NVIDIA Container Runtime for easy deployment
- **NGC Integration**: Access to NVIDIA's container registry

Important

Use of the NVIDIA CUDA Toolkit preinstalled on DGX Spark is subject to the NVIDIA CUDA Toolkit End User License Agreement.

For detailed software information, see *Software*.

2.5.1.5 Getting Started

To begin using your DGX Spark:

1. **Initial Setup**: Follow the *Initial Setup - First Boot* to configure your system
2. **Explore Examples**: Try sample workloads to understand capabilities
3. **Configure Development Environment**: Set up your preferred tools and frameworks
4. **Start Building**: Begin your AI development projects

Note

For the most up-to-date tutorials and examples, visit https://build.nvidia.com/spark. This site is regularly updated with new content and serves as the primary resource for practical guides and use cases.

2.5.2. UEFI Settings

This topic provides guidance on accessing and configuring the UEFI settings for the DGX Spark system. While there are no Spark-specific features that require UEFI configuration, you may need to access the UEFI for general system configuration or troubleshooting purposes.

2.5.2.1 Access UEFI

Important

To access the UEFI setup menu, you must be using a keyboard that is physically connected to the DGX Spark device.

If the keyboard is a Mac keyboard, it might not have a key that UEFI recognizes as Del; use Esc only.

To access the UEFI setup menu:

1. Power on or restart the system.
2. Immediately press `Esc` or `Del` and hold it until the UEFI setup menu appears.

Note

The timing for pressing `Esc` or `Del` is critical. Press the key as soon as the system starts booting, before the OS begins loading.

2.5.2.2 UEFI Documentation

For detailed information about UEFI settings and configuration options, refer to the DGX Spark UEFI Manual.

2.5.2.3 Enable or Disable WiFi and Bluetooth

You can enable or disable both Wireless LAN and Bluetooth together in UEFI, which completely disables all wireless access for security enhancement. To disable only one (WiFi or Bluetooth) individually, use the OS settings instead. You might disable wireless to meet lab or security policy, or enable it for initial setup when Ethernet is not available.

1. Go to **Advanced** → **Advanced Menu** → **IO Port Access**.
2. Set **Wireless LAN & Bluetooth** to the desired state (enabled or disabled).

2.5.2.4 Change Display Reserved Memory

DGX Spark reserves 2 GB of system memory for the display by default. If you use DGX Spark without a display (headless), you can make more memory available to applications.

1. Run `free -h` and record the total memory.
2. Restart DGX Spark.
3. Immediately press `Esc` or `Del` and hold it until the UEFI setup menu appears.
4. Go to **Advanced** → **Advanced Menu** → **Display Reserved Memory**.
5. Select one of these options:
   - **Auto:** UEFI sizes the reservation from the displays that are connected at boot. A headless system reserves 0.5 GB.
   - **0.5 GB**, **1 GB**, **2 GB**, or **4 GB:** UEFI reserves the selected amount.
6. Go to **Save & Exit** and select **Save Changes and Reset**.
7. After DGX Spark restarts, run `free -h` again and compare the total memory with the value you recorded.

For more information, refer to Display Reserved Memory in the DGX Spark UEFI Manual.

2.5.2.5 Boot from a USB Device

To boot from a USB device, use one of the following methods. You might do this to run system recovery, reinstall the OS, or boot from live or diagnostic media.

Set USB as the first boot device (persistent):

1. Go to **Boot** → **Boot Option Priorities**.
2. Use `Enter` on each entry to set the boot order. Place the USB device at the top.

Boot from USB once (one-time):

1. Go to **Save & Exit**.
2. Under **Boot Override**, select the USB device and press `Enter`. The system boots from that device for the current boot only.

The USB device must appear in the boot option list. If it does not, ensure the device is connected before you enter UEFI and that it is bootable.

2.5.2.6 Enable or Disable Secure Boot

To enable or disable Secure Boot, perform the following steps. You might disable it to use PXE boot or run custom boot loaders, or enable it to meet security or compliance requirements.

1. Go to **Security** → **Secure Boot**.
2. Set **Secure Boot** to **Enabled** or **Disabled** as needed. The default is **Enabled**.
3. Save changes and exit from the **Save & Exit** menu. A platform reset may be required for the change to take effect.

Secure Boot is active when it is enabled, the Platform Key (PK) is enrolled, and the system is in User mode.

2.5.2.7 Configure PXE Boot

To configure UEFI for PXE boot, perform the following steps. You might do this to deploy or reimage the DGX Spark over the network, or to boot the recovery image from a PXE server.

> **Note**
>
> Before enabling PXE boot, you must either disable Secure Boot or enroll the PXE bootloader (`grubnetaa64.efi.signed`) in UEFI. For details, see *PXE Boot Setup*.

1. Go to **Advanced** → **Network Stack Configuration**.
2. Set **Network Stack** to **Enabled**.
3. Set **Ipv4 PXE Support** or **Ipv6 PXE Support** (or both) to **Enabled** as needed.
4. Optionally set **PXE boot wait time** (seconds to press `Esc` to abort PXE) and **Media detect count**.
5. In **Advanced**, open the network configuration screen for the NIC you want to use for PXE (listed by MAC address). Enable and configure IPv4 or IPv6 for that device as required.
6. Save changes and exit from the **Save & Exit** menu.

For PXE server setup, bootloader configuration, and DHCP setup, see *PXE Boot Setup*.

Important

Always save your UEFI configuration changes from the **Save & Exit** menu so the system restarts with your new settings applied.

2.5.2.8 Troubleshooting

If you experience issues after making UEFI changes:

- Revert to UEFI defaults and restart
- Apply changes incrementally to identify problematic settings
- Update to the latest UEFI version if available
- Consult the DGX Spark UEFI Manual for specific setting descriptions
- For support options, see *Get the Right Support for Your DGX Spark*.

2.5.3. ConnectX-7 Networking

2.5.3.1 Connecting DGX Spark Systems into a Cluster

2.5.3.1.1 Overview

You can connect multiple DGX Spark systems with cables to create a cluster that allows you to run workloads that cannot fit onto a single device.

At the highest level, clustering DGX Spark devices requires you to:

1. Physically connect them with Quad Small Form-factor Pluggable (QSFP) cables plugged into the external QSFP ports.
2. Detect the network interfaces that need IP addresses for inter-device communication.
3. Set up the IP layer to route traffic to those interfaces.

Step one is straightforward, as described in *QSFP Ports and Cables*. However, step two can be confusing if you are unfamiliar with the unique PCIe topology of the DGX Spark. That confusion can lead to mistakes in step three and prevent the cluster from working correctly.

This section introduces the details needed to understand the basics of clustering the devices without doing a full PCIe deep dive.

If you just need a quick reference, skip to the *Full Correspondence Table*.

2.5.3.1.2 The QSFP Ports and Cables

QSFP technology is a compact, hot-swappable, and bidirectional transceiver for high-speed data transfer. Each DGX Spark has two QSFP ports (sometimes called “ConnectX-7 ports”) on the back of the device. Each port provides up to 200 Gigabits per second (Gb/s), but the incoming speed is also determined by the cable that you use.

QSFP cables come in different generations and speeds:

- Use cables that are known to provide at least 200 Gb/s.
- Using a cable with higher speed is not beneficial, because the port itself is capped at 200 Gb/s.

Note

The QSFP ports support Ethernet configuration only. Approved cables are:

- Amphenol: NJAAKK-N911 (QSFP to QSFP112, 32AWG, 400mm, LSZH), *NJAAKK0006 is the 0.5m version of this cable*
- Luxshare: LMTQF022-SD-R (QSFP112 400G DAC Cable, 400mm, 30AWG)

2.5.3.1.2.1 Plugging in a QSFP Cable

When it comes to plugging in the cables, the QSFP ports are interchangeable. However, the current clustering playbooks (Connect Two Sparks, Connect Three Sparks, and Multi Sparks Through a Switch) assume certain ports. Therefore, it is important to know the left port from the right port.

When seen from the back of the device, the left port is the QSFP port closest to the ethernet port. To contrast this, in the figure below the cable is plugged into the **right** port.

To properly plug a QSFP cable into a port, complete the following steps:

1. Turn the DGX Spark so that the back is facing you with the ports exposed.
2. Select one of the two QSFP ports.
3. Orient the cable so the pull-tab (sometimes called a “ring tab”) faces upward. This is shown in the right port of the figure below.
4. Plug the cable into the selected port. It should slide in smoothly without force. If that is not happening, make sure the pull-tab is facing up.
5. Make sure the cable is fully inserted.



Warning

Do not force the QSFP cable into the port. If it does not slide in smoothly, stop, verify tab orientation and port alignment, and try again. Forcing an upside-down or misaligned connector can damage the port.

2.5.3.1.2.2 Removing the QSFP Cable

You can remove the cable by pulling the pull-tab. Its should come out easily.

2.5.3.1.3 Understand the ConnectX-7 Network

The QSFP ports are connected to the high-bandwidth ConnectX-7 network interface controller (NIC) in the DGX Spark. The NIC is the heart of high-speed inter-device communication between connected DGX Spark systems.

Keep the following high-level facts in mind:

- The NIC sits between the external QSFP ports and the Grace Blackwell system on a chip (SoC).
- The NIC connects independently to the two external QSFP ports, and it connects to the SoC through two independent PCIe Gen 5 x4 links.
- As a result, each QSFP port has two PCIe addresses to account for the two PCIe x4 links out of the NIC into the SoC.

These facts imply the following Linux network configuration details:

- Each QSFP port appears as two independent Linux Ethernet interfaces. As a result, plugging in two cables shows a total of four Linux Ethernet interfaces.
- Each Ethernet interface has a corresponding RoCE interface (typically called a "RoCE device") for InfiniBand communication.

2.5.3.1.3.1 Interface Names and PCIe Addresses

When you plug a QSFP cable into the back of a DGX Spark, Linux automatically creates four network interface names in order to give the two PCIe addresses in the port their own unique Ethernet and RoCe names. If you plug in two cables, then you double the number to get a total of eight unique network interfaces.

Linux builds the names from the PCIe addresses, and so the interface names are the same on all DGX Spark devices. However, they are difficult to understand without knowing the naming pattern. This section explains the pattern using the `ibdev2netdev` command for a DGX Spark with cables in both QSFP ports.

The following output shows the eight network interface names for the two QSFP ports, with four RoCE on the left and four Ethernet on the right:

```
nvidia@spark-1afa:~$ ibdev2netdev
rocep1s0f0   port 1 ==> enp1s0f0np0   (Up)
rocep1s0f1   port 1 ==> enp1s0f1np1   (Up)
roceP2p1s0f0 port 1 ==> enP2p1s0f0np0 (Up)
roceP2p1s0f1 port 1 ==> enP2p1s0f1np1 (Up)
```

Looking across a single row, you can see that the central characters of both names are a shared PCIe address. For example, both sides of row 1 share `p1s0f0`, and both sides of row 3 share `P2p1s0f0`.

The other characters in the names are just there to differentiate RoCe versus Ethernet, as well as specify the interface ports. For example:

- `roce` (RoCE) on the left and `en` ("Ethernet") distinguish the types.
- On the Ethernet side, `np0` and `np1` designate the Ethernet network ports.
- On the RoCE side, `port 1` designates the RoCE port.

Differentiating the PCIe addresses is a bit more complicated but can be done by splitting them into individual fields as show below:

Table 3: PCIe Address Fields from ibdev2netdev Output

Row	PCIe Domain	Bus	Slot	Function
1	blank	p1	s0	f0
2	blank	p1	s0	f1
3	P2	p1	s0	f0
4	P2	p1	s0	f1

Note the following about this table of split PCIe addresses:

1. The PCIe domain of rows 1 and 2 is actually `0`, but it is omitted by convention and so is blank.
2. The PCIe domain of rows 3 and 4 is 2. It is included to avoid conflation with the previous convention for `0`.

3. Moving left to right specifies the address corresponding to outputs of the `lspci` command.

With this vocabulary and convention, you can read row 1 of the `ibdev2netdev` command as follows:

"The RoCE interface at PCIe domain 0, bus 1, slot 0, function 0, and RoCE port 1 corresponds to the Ethernet interface at PCIe domain 0, bus 1, slot 0, function 0, and network port 0."

A similar statement for row 4 of the command output is:

"The RoCE interface at PCIe domain 2, bus 1, slot 0, function 1, and RoCE port 1 corresponds to the Ethernet interface at PCIe domain 2, bus 1, slot 0, function 1, and network port 1."

2.5.3.1.3.2 Full Correspondence Table

The following table shows the correspondences between the QSFP ports and relevant addresses:

Table 4: QSFP Port to PCIe, Ethernet, and RoCE Address Mapping

QSFP Port	Port #	PCIe Address	Ethernet Address	RoCE Address
Left	0	p1s0f0	enp1s0f0np0	rocep1s0f0
Left	0	P2p1s0f0	enP2p1s0f0np0	roceP2p1s0f0
Right	1	p1s0f1	enp1s0f1np1	rocep1s0f1
Right	1	P2p1s0f1	enP2p1s0f1np1	roceP2p1s0f1

2.5.3.1.4 Next Steps

2.5.3.1.4.1 Cluster with NVIDIA Sync

The *NVIDIA Sync* Cluster Assistant provides a guided way to configure a supported DGX Spark cluster. It does automatic discovery and network creation to eliminate the network configuration you would typically do manually in a terminal.

The Cluster Assistant validates the devices, applies ConnectX-7 network settings, checks link performance, and configures SSH between nodes. It supports up to three DGX Spark systems connected directly through cables, and up to four systems when using a switch.

For NVIDIA Sync installation and the Cluster Assistant workflow, refer to the NVIDIA Sync User Guide.

2.5.3.1.4.2 Playbooks to Connect Multiple DGX Sparks

For manual configuration using playbooks, refer to the following connection options:

- Connect Two Sparks
- Connect Three Sparks
- Multi Sparks Through a Switch

2.6. Software

The DGX Spark comes with a comprehensive software stack optimized for AI development, machine learning, and data science workflows. This section provides detailed information about the included

software components and their configuration.

Important

Use of the NVIDIA CUDA Toolkit preinstalled on DGX Spark is subject to the NVIDIA CUDA Toolkit End User License Agreement.

2.6.1. DGX Dashboard

2.6.1.1 Overview

The DGX Spark comes with a built-in dashboard that provides an overview of the system's current operational metrics, the ability to apply updates, change some system settings, and access local Jupyter Notebooks.

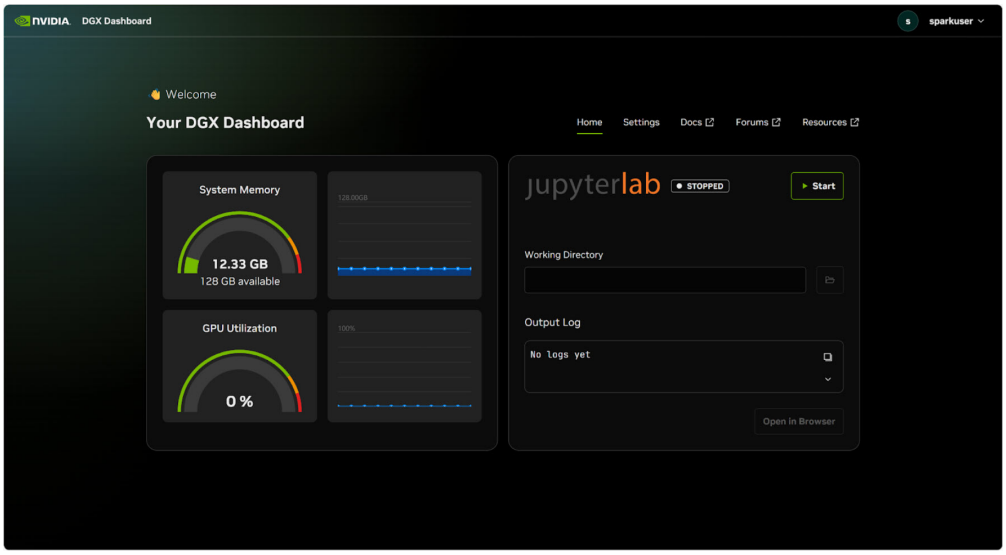


Fig. 2: The DGX Dashboard provides real-time system monitoring and integrated JupyterLab access

Note

To run updates and change the device name, you must have **sudo** access. The account created during initial setup will have access.

If your administrator has disabled updates, the dashboard shows the "Updates Disabled" state instead, regardless of your **sudo** access. Refer to *Administrator Policy Settings* for more information.

2.6.1.2 Integrated JupyterLab

The dashboard includes an integrated JupyterLab instance that provides a convenient development environment:

- When started, JupyterLab creates a virtual environment in the specified working directory and automatically installs a set of recommended packages
- If you enter a new working directory and start JupyterLab, a new environment will be created

- Each user account on the device is assigned a port located in `/opt/nvidia/dgx-dashboard-service/jupyterlab_ports.yaml`
- To access JupyterLab remotely, you must tunnel it just like the dashboard itself. The port to tunnel is in the ports file. Using *NVIDIA Sync*, this tunnel is managed for you automatically and just works

2.6.1.3 Accessing the Dashboard

The dashboard can be accessed locally by clicking on the "Show Apps" button in the bottom left corner of the Ubuntu desktop. Then, in the app grid, select the "DGX Dashboard" shortcut to open the dashboard in your default web browser.

Remotely, the dashboard can be accessed using *NVIDIA Sync* or through a manually created SSH tunnel. If using NVIDIA Sync, after connecting, simply click on the "DGX Dashboard" button and the dashboard will open in your default web browser at http://localhost:11000.

To manually access over SSH, first open a tunnel, e.g., `ssh -L 11000:localhost:11000 <username>@<IP or spark-abcd.local>`. Then, open the dashboard in your web browser at http://localhost:11000.

2.6.1.4 Administrator Policy Settings

Administrators can restrict which dashboard features are available to users. This is intended for deployments where administrators manage updates and do not want users to start updates themselves.

The dashboard reads these restrictions from a root-owned JSON file at `/opt/nvidia/dgx-dashboard/settings.json`.

2.6.1.4.1 Disabling Dashboard Updates

To prevent users from installing updates through the dashboard:

1. Create `/opt/nvidia/dgx-dashboard/settings.json` with the following content:

```
{
  "update": {
    "enabled": false
  }
}
```

2. Restart the dashboard service to apply the change immediately:

```
sudo systemctl restart dgx-dashboard.service
```

   If you do not restart the service, the dashboard applies the change within 10 minutes.

The dashboard then shows the "Updates Disabled" state in place of the update controls described in *Using DGX Dashboard for Updates*.

2.6.1.4.2 Supported Policy Fields

Field	Description
`update.enabled`	Enables or disables the dashboard update interface and update actions. Defaults to `true`.
`wifiAssistant.enabled`	Allows or disallows the Wi-Fi Assistant.

The dashboard reads these settings only from `settings.json`. If Landscape manages this file, the dashboard disables update operations and shows the “Updates Disabled” state. Disabling updates in Landscape through any other mechanism does not change the dashboard.

2.6.2. NVIDIA Sync

2.6.2.1 Overview

NVIDIA Sync is a utility for Windows, macOS, and Ubuntu that streamlines access to your DGX Spark from a laptop or workstation.

It manages SSH connections, port forwarding, and tunnels in the background so you can work on your DGX Spark without manually running commands in a terminal.

- Learn more in the NVIDIA Sync User Guide.
- Download and install NVIDIA Sync.

2.6.2.2 Key Capabilities

- **Direct connections**: Add a device on your local network using device discovery, a hostname or IP address, or an imported SSH alias. Refer to Direct Connections in the NVIDIA Sync User Guide.
- **Tailscale connections**: Access your DGX Spark when you are not on the same network. Refer to Tailscale Connections in the NVIDIA Sync User Guide.
- **Applications**: Launch supported IDEs, tools, and custom scripts on your DGX Spark with a click. Refer to Applications in the NVIDIA Sync User Guide.
- **Cluster Assistant**: Cluster your DGX Spark systems without manually running commands in a terminal. Refer to Cluster Assistant in the NVIDIA Sync User Guide. For physical cable requirements and ConnectX-7 network background, refer to *ConnectX-7 Networking*.

2.6.2.3 Using NVIDIA Sync with Your DGX Spark

To get started with NVIDIA Sync:

1. Install NVIDIA Sync on your laptop.
2. Add your DGX Spark to NVIDIA Sync.
3. Launch an application on your DGX Spark through NVIDIA Sync.

For step-by-step installation and onboarding instructions, refer to Getting Started in the NVIDIA Sync User Guide.

2.6.2.4 DGX Spark Skills

Skills give agents reusable guidance for specific tasks. DGX Spark Skills provide DGX Spark-specific knowledge and workflows for troubleshooting, clustering multiple DGX Spark systems, device maintenance, playbook discovery, support escalation, and searches of official documentation and user forums.

Starting in late October 2026, you can access these skills through the local OpenCode instance and web interface that NVIDIA Sync configures.

To implement the skills in an existing agent, install them from the NVIDIA agent skills catalog, under `dgx-spark-skills`. The catalog includes the `npx` installation instructions.

2.6.3. DGX OS

2.6.3.1 Overview

NVIDIA DGX OS is a customized Linux distribution that provides a stable, tested, and supported operating system foundation for running AI, machine learning, and analytics applications on DGX systems. It includes platform-specific optimizations, drivers, and diagnostic tools tailored for NVIDIA hardware.

DGX OS serves as the underlying operating system for your DGX Spark, providing:

- A robust Linux foundation optimized for AI workloads
- Pre-configured drivers and system settings for NVIDIA hardware
- Security updates and system maintenance capabilities
- Compatibility with the broader NVIDIA software ecosystem

Important

Reinstalling or recovering your DGX Spark: DGX Spark uses a different recovery process than enterprise DGX systems. Do not use the Enterprise Download Center or DGX OS ISO. For additional instructions, refer to *System Recovery*. The recovery image is available from developer.nvidia.com and does not require an enterprise account.

Note

For more information about DGX OS, see the official documentation at: https://docs.nvidia.com/dgx/dgx-os-7-user-guide/introduction.html

2.6.3.2 Security and Compliance

DGX OS is based on Ubuntu. For Ubuntu security capabilities and supported compliance standards (such as FIPS, CIS, and DISA-STIG), see the Ubuntu security standards.

2.6.3.3 Release Cadence

DGX OS follows a regular release schedule with updates typically provided twice per year, around February and August, for the first two years after initial release. Additional updates and security patches are provided between major releases and throughout the support lifecycle.

2.6.4. NVIDIA Nsight

2.6.4.1 Overview

NVIDIA Nsight is a family of GPU profiling, debugging, and analysis tools for CUDA and graphics workloads. Individual tools focus on different tasks in the optimization workflow:

- **Nsight Systems** — Timeline profiling of CPU, GPU, and system activity for a full application run.
- **Nsight Graphics** — Frame capture and pipeline inspection for graphics APIs.
- **Nsight DL Designer** — Visualization and analysis for deep learning models.
- **Nsight Compute** — CUDA kernel profiling, memory metrics, and guided performance reports.

The DGX Spark stack follows the broader NVIDIA software release model. Current installers, release notes, supported platforms, and user documentation for each Nsight tool are maintained on the NVIDIA Developer website. Use the pages in the following table to retrieve the latest build for your development system.

2.6.4.2 Latest Versions

Tool	NVIDIA Developer Page
Nsight Systems	Get started with Nsight Systems
Nsight Graphics	Get started with Nsight Graphics
Nsight DL Designer	Get started with Nsight DL Designer
Nsight Compute	Get started with Nsight Compute

2.6.5. NVIDIA Container Runtime for Docker

2.6.5.1 Overview

The NVIDIA Container Runtime enables Docker containers to access GPU resources on DGX Spark systems. It provides hooks based on the Open Container Initiative (OCI) specification, which is an open standard for container runtimes. These OCI hooks are triggered when the `--gpus` flag is used, ensuring GPU devices are made available inside the container. The runtime acts as a bridge between Docker and the NVIDIA drivers, allowing containers to utilize GPU acceleration for AI/ML workloads, CUDA applications, and other GPU-accelerated software.

Key benefits: - Seamless GPU access within containers - Automatic driver and library management - Support for multi-GPU configurations - Compatibility with popular container orchestration platforms

The runtime works in conjunction with the NVIDIA Container Toolkit, which provides the necessary components to expose GPU devices and CUDA libraries to containerized applications.

2.6.5.2 Installation

The NVIDIA Container Toolkit is preinstalled and configured on DGX Spark systems. This includes:

- NVIDIA Container Runtime
- Docker integration
- GPU device access configuration

- CUDA library management

The runtime is ready to use out of the box for running GPU-accelerated containers.

2.6.5.2.1 Optional: Add User to Docker Group

By default, Docker requires `sudo` privileges to run commands. Adding your user to the docker group allows you to run Docker commands without `sudo`, which provides:

- **Convenience**: No need to type *sudo* before every Docker command
- **Better workflow**: Seamless integration with development tools and scripts
- **Reduced friction**: Faster iteration when working with containers

To add your user to the docker group:

```
sudo usermod -aG docker $USER
newgrp docker
```

> **Note**
>
> This step is optional. You can continue using Docker with `sudo` if you prefer not to modify group memberships.

2.6.5.3 Usage

2.6.5.3.1 Basic GPU Access

Run a container with GPU access using the `--gpus` flag:

```
docker run -it --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.04 nvidia-
↪smi
```

This command: - Runs an interactive container (`-it`) - Enables access to all GPUs (`--gpus=all`) - Uses the NVIDIA CUDA development image - Executes `nvidia-smi` to display GPU information

2.6.5.3.2 Set GPU Capabilities

Control which GPU capabilities are available to the container:

```
docker run -it --gpus '"capabilities=compute,utility"' nvcr.io/nvidia/cuda:13.
↪0.1-devel-ubuntu24.04 nvidia-smi
```

2.6.5.4 Validation

2.6.5.4.1 Test GPU Access

Run the test command to verify GPU access:

```
docker run -it --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.04 nvidia-
↪smi
```

Expected output should show:

- GPU device information

- Driver version
- CUDA version
- Memory usage and temperature

2.6.5.5 Troubleshooting

2.6.5.5.1 Runtime Not Found

If you encounter “runtime not found” errors:

1. Verify NVIDIA Container Toolkit is installed:

```
nvidia-ctk --version
```

2. Check Docker daemon configuration:

```
cat /etc/docker/daemon.json
```

3. Restart Docker service:

```
sudo systemctl restart docker
```

2.6.5.5.2 Driver/Container CUDA Mismatch

If you see CUDA version mismatches:

1. Check host CUDA driver version:

```
nvidia-smi
```

2. Use a container image with compatible CUDA version:

```
docker run -it --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.04
↪nvidia-smi
```

2.6.5.5.3 Permission Issues

If you encounter permission errors:

1. Ensure your user is in the docker group (if not using sudo):

```
groups $USER
```

2. Check device permissions:

```
ls -la /dev/nvidia*
```

3. Verify Docker daemon has access to GPU devices:

```
sudo docker run -it --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.
↪04 nvidia-smi
```

2.6.5.5.4 Container Startup Issues

If containers fail to start:

1. Check Docker logs:

```
docker logs <container_id>
```

2. Verify GPU devices are available on host:

```
ls /dev/nvidia*
```

3. Test with a minimal container:

```
docker run --rm --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.04
↪echo "GPU test successful"
```

2.6.6. NGC

2.6.6.1 Overview

NVIDIA GPU Cloud (NGC) is a comprehensive registry of GPU-optimized containers, pre-trained models, and AI/ML software that enables rapid development and deployment of AI applications. For DGX Spark users, NGC provides access to the latest frameworks, tools, and optimized environments specifically designed for the Grace Blackwell architecture.

Key benefits for DGX Spark users:

- **Optimized Containers**: Pre-configured environments with the latest AI/ML frameworks, CUDA, and libraries optimized for Grace Blackwell GPUs
- **Pre-trained Models**: Access to state-of-the-art models and model collections for various AI tasks
- **Rapid Development**: Skip complex environment setup and focus on your AI/ML projects
- **Cutting-edge Software**: Access to the latest NVIDIA software stack and experimental features

NGC is particularly valuable for DGX Spark users because it provides the most current and optimized software stack for this new platform, ensuring you have access to the latest performance optimizations and features.

2.6.6.2 Access NVIDIA NIM Containers

NVIDIA NIM microservices (NIM) containers on NGC require authenticated NGC access. For research, development, and experimentation, join the free NVIDIA Developer Program and generate an NGC Personal API Key on the Setup API Keys page. When creating the Personal API Key, select at least **NGC Catalog** from the **Services Included** list. Legacy API keys are not supported by NVIDIA NIM for LLMs.

For production use, enterprise support, or supported software lifecycle requirements, obtain or activate an NVIDIA AI Enterprise license.

DGX Spark and GB10-based OEM systems do not use device serial-number registration to unlock NIM container pulls. Use an NGC Personal API Key with the NGC Catalog service enabled, then authenticate to `nvcr.io` with username `$oauthtoken`.

Before pulling a NIM container, confirm that the model has a DGX Spark-compatible NIM image or profile in the DGX Spark Collection or the NVIDIA NIM for LLMs support matrix. Not every NIM has a DGX Spark variant.

For step-by-step NIM instructions, see Get Started with NVIDIA NIM for LLMs and the NIM on Spark playbook.

2.6.6.3 Getting Started

2.6.6.3.1 Create an NGC Account

1. Visit the NGC website
2. Click **Sign Up** and create a free account
3. Verify your email address
4. Complete your profile information

2.6.6.3.2 Generate an API Key

1. Log in to your NGC account
2. Navigate to **Setup** -> **API Key**
3. Click **Generate API Key**
4. Copy and securely store your API key

> **Note**
>
> Your API key is required for pulling containers and accessing NGC resources. Keep it secure and never share it publicly.

2.6.6.3.3 Install NGC CLI (Optional)

The NGC CLI provides convenient command-line access to NGC resources. Your DGX Spark system requires the ARM64 version of the NGC CLI which can be found on the **ARM64 Linux** tab at https://org.ngc.nvidia.com/setup/installers/cli.

For more information on installing and using the NGC CLI, see NGC CLI Documentation.

2.6.6.3.4 Authenticate with Docker

Configure Docker to access NGC registries:

```
# Login to NGC with Docker
docker login nvcr.io
# Username: $oauthtoken
# Password: <your-api-key>
```

2.6.6.4 Basic Usage

2.6.6.4.1 Pull and Run a Container

Start with a popular AI/ML framework container:

```
# Pull a PyTorch container optimized for Grace Blackwell
docker pull nvcr.io/nvidia/pytorch:24.08-py3

# Run the container with GPU access
docker run -it --gpus=all nvcr.io/nvidia/pytorch:24.08-py3
```

2.6.6.4.2 Explore Available Resources

Browse NGC resources through the web interface:

- **Containers**: AI/ML frameworks, development environments, and specialized tools
- **Models**: Pre-trained models for computer vision, natural language processing, and more
- **Helm Charts**: Kubernetes deployment configurations
- **Jupyter Notebooks**: Interactive tutorials and examples

2.6.6.5 Common Workflows

2.6.6.5.1 Development Environment

Use NGC containers as your development environment:

```
# Run a development container with persistent storage
docker run -it --gpus=all \
  -v /path/to/your/project:/workspace \
  nvcr.io/nvidia/pytorch:24.08-py3
```

2.6.6.5.2 Model Inference and Training

Access pre-trained models and training scripts:

```
# Pull a model from NGC
ngc registry model download-version "nvidia/nemo/bertbaseuncased:1.0.0rc1"

# Or use models directly in containers
docker run -it --gpus=all \
  nvcr.io/nvidia/pytorch:24.08-py3
```

2.6.6.6 Best Practices

2.6.6.6.1 Container Management

- **Pin Versions**: Use specific container tags for reproducible environments
- **Regular Updates**: Periodically update to newer container versions for latest optimizations
- **Resource Limits**: Set appropriate memory and CPU limits for your workloads

2.6.6.6.2 Data Persistence

- **Volume Mounts**: Mount your data directories into containers for persistence
- **Model Storage**: Store trained models and checkpoints outside containers

- **Configuration**: Keep configuration files in version control

2.6.6.6.3 Security

- **API Key Security**: Store your NGC API key securely and rotate it regularly
- **Container Scanning**: Scan containers for vulnerabilities before use
- **Network Security**: Use appropriate network configurations for your environment

2.6.6.7 Troubleshooting

2.6.6.7.1 Common Issues

Authentication Failures

```
# Verify your API key is correct
docker login nvcr.io
# Check if your account has access to the requested resource
```

Container Pull Issues

```
# Check network connectivity to NGC registry
curl -I https://ngc.nvidia.com

# View container catalog on NGC website
# Visit https://catalog.ngc.nvidia.com/containers

# Or use NGC CLI to list available containers
ngc registry image list nvidia/*

# Try pulling with verbose output to see specific error
docker pull nvcr.io/nvidia/pytorch:24.08-py3
```

GPU Access Problems

```
# Verify NVIDIA Container Runtime is installed
docker run --rm --gpus=all nvcr.io/nvidia/cuda:13.0.1-devel-ubuntu24.04
↪nvidia-smi
```

2.6.6.7.2 Getting Help

- **NGC Documentation**: Visit the NGC documentation
- **Community Forums**: Join the NVIDIA Developer Forums
- **Additional Support**: For support options, see *Get the Right Support for Your DGX Spark*

2.6.7. NVIDIA AI Enterprise—DGX Spark Quick Start Guide

NVIDIA AI Enterprise—DGX Spark is an enterprise-grade software platform for AI development, deployment, and optimization on DGX Spark and NVIDIA GB10 Grace Blackwell Superchip-based systems. It provides libraries, frameworks, microservices, drivers, and enterprise support to accelerate inference, training, customization, observability, and guardrailing.

2.6.7.1 Set Up Your DGX Spark System

Before accessing NVIDIA AI Enterprise—DGX Spark, you must complete the initial setup of your DGX Spark system. Follow the steps in *First Boot and Initial Configuration*. After your system is powered on and connected to your network, continue with the steps in this guide.

2.6.7.2 NIM Access and NVIDIA AI Enterprise Entitlement

NVIDIA NIM microservices (NIM) container access and NVIDIA AI Enterprise—DGX Spark entitlement are related, but they are not the same thing. You can access eligible NVIDIA NIM containers through NVIDIA Developer Program membership or through NVIDIA AI Enterprise. An NVIDIA AI Enterprise—DGX Spark entitlement is available only if you purchased it, requested an evaluation, or received an NVIDIA Entitlement Certificate that lists NVIDIA AI Enterprise—DGX Spark.

If you do not have an NVIDIA Entitlement Certificate for NVIDIA AI Enterprise—DGX Spark, do not use the generic NVIDIA AI Enterprise GPU serial-number activation workflow to resolve NIM pull failures. Instead, use an NGC Personal API key to authenticate to `nvcr.io`, then pull and run the NIM container by following Get Started with NVIDIA NIM for LLMs or the NIM on Spark playbook. If your entitlement certificate instructs you to activate a license and activation returns an “on hold” or “already claimed” message, contact NVIDIA Enterprise Support and include the activation reference ID.

2.6.7.3 Optional: NVIDIA AI Enterprise—DGX Spark Evaluation

If you would like to try NVIDIA AI Enterprise—DGX Spark before purchasing, NVIDIA offers a 90-day trial of NVIDIA AI Enterprise. You can register here.

2.6.7.4 Order Confirmation Message

After your order is processed, you will receive an order confirmation message from NVIDIA containing the information required to access NVIDIA AI Enterprise software and enterprise support.

Your NVIDIA Entitlement Certificate is attached to the message and provides instructions for using the certificate.

If you manage software access or licensing in your organization, follow the instructions in the NVIDIA Entitlement Certificate. Otherwise, forward the order confirmation message and NVIDIA Entitlement Certificate to your IT administrator.

2.6.7.5 Create Your NVIDIA AI Enterprise Account

To access NVIDIA AI Enterprise—DGX Spark and technical support, you must have an NVIDIA Cloud Account (NVIDIA AI Enterprise Account), which provides access to:

- **NVIDIA NGC** - Access to NVIDIA AI Enterprise—DGX Spark software component.
- **NVIDIA Enterprise Support Portal** - Access to support services.

Both can be accessed from the NVIDIA Application Hub.

Before you begin, have your order confirmation message available.

1. In the NVIDIA Entitlement Certificate instructions, follow the **Register** link.
2. Fill out the form on the NVIDIA Enterprise Account Registration page and click **REGISTER**. You will receive an email with instructions to log in to the NVIDIA Application Hub.
3. Open the email and click **Log In**.
4. On the Login page, enter your email address and click **Sign In**.

5. On the Create Your Account page, provide and confirm a password, then click **Create Account**. You will receive an email to verify your address.
6. Open the verification email and click **Verify Email Address**.

You can now log in to the NVIDIA Application Hub to access NVIDIA NGC and the NVIDIA Enterprise Support Portal.

2.6.7.6 Access NVIDIA AI Enterprise—DGX Spark Software Asset

1. Go to the NVIDIA NGC Catalog and log in if prompted.
2. On the **NVIDIA NGC Search** page, set the **DGX Spark** filter or look for the **DGX Spark Collection**.
3. Click the software asset to learn more or download it.

2.7. Common Use Cases

The DGX Spark is designed to support a wide range of AI, machine learning, and data science workflows. Visit https://build.nvidia.com/spark for practical guides to help you get started. That site is regularly updated with new content and information and will be the single source of truth for your Spark device.

2.8. PXE Boot Setup

The DGX Spark UEFI BIOS supports PXE boot. Several manual customization steps are required to get PXE to boot the DGX OS image or the DGX Spark recovery image.

Caution

This document is meant to be used as a reference. Explicit instructions are not given to configure the DHCP, HTTP, and TFTP servers. The end user's IT team is expected to configure these servers to fit their company's environment and security guidelines.

2.8.1. Requirements

- TFTP server
  - Software that provides TFTP service.
- HTTP server
  - An HTTP server is used to transfer large files, such as the iso image and `initrd`. alternatively, TFTP can be used for this purpose. HTTP is used in the example below.
- DHCP server
  - Software that provides dynamic host configuration protocol (DHCP) service.

> **Note**
>
> The TFTP server, HTTP server, and DHCP server can all be configured on the same system, or they can each be on different systems.

- linux bootloader
- ip address: `<ftp ip>`
- fully qualified host: `<ftp host>`

This topic provides some guidance concerning how to set up a PXE boot environment for DGX systems. For complete details, refer to online documentation for setting up a PXE boot server. In this example, `xinetd` is used to provide TFTP service; `dnsmasq` is used to provide DHCP service; and `syslinux` is used as the bootloader.

2.8.2. Overview of the PXE Server

The PXE server requires configuration in the following areas:

- bootloader (grub)
- TFTP contents (the kernel and `initrd`)

  In this example, TFTP is configured to serve files from `/local/tftp/`. You will need to configure your TFTP server to serve files from `/local/tftp` or the directory you desire to use.
- HTTP contents (the iso image)

  In this example, HTTP is configured to serve files from `/local/http/`. You will need to configure your HTTP server to serve files from `/local/http` or the directory you desire to use.
- DHCP

2.8.2.1 PXE Server Configuration for DGX OS Image

In this example, the directory structure on the HTTP and TFTP server looks like this:

```
/local/
   http/
      base_os_7.0.0/
         base_os_7.0.0.iso

   tftp/
      baseos/
         vmlinuz
         initrd
      grub/
         grub.cfg
      grubnetaa64.efi.signed
```

> **Note**
>
> The `vmlinuz` and `initrd` files are specified relative to the TFTP root, `/local/tftp/`; and the location of the ISO, `base_os_7.0.0.iso`, is relative to the HTTP root, `/local/http/`.

Here, the DHCP and PXE servers are configured to use the above directory structure. The person responsible for deploying the PXE environment should change the directory names and structure to fit their infrastructure.

You can set up the directory structure on your HTTP and TFTP server similarly.

The contents of the /local/tftp/grub/grub.cfg file should look something like this:

```
set default=0
set timeout=5

menuentry 'Install BaseOS 7.0.0' {
   linux /baseos/vmlinuz fsck.mode=skip autoinstall ip=dhcp url=http://
↪<Server IP>/base_os_7.0.0/base_os_7.0.0.iso nvme-core.multipath=n nouveau.
↪modeset=0
   initrd /baseos/initrd
}
```

Note

The kernel boot parameters should match the contents of the corresponding ISO's boot menu found in `/mnt/boot/grub/grub.cfg`.

When the system being installed boots via PXE, boot files located on `/local/tftp` are retrieved from the TFTP server. (In this example, the TFTP server is provided by the `xinetd` service whose configuration file, `/etc/xinetd.d/tftp`, specifies that the boot files are located on `/local/tftp`.) When a system is PXE booted, the `grubnetaa64.efi.signedi` file that is designated in the DHCP server's `dhcpd.conf` file is retrieved by TFTP transfer (see *Configure Your DHCP Server*). By default, after the `grubnetaa64.efi.signed` is booted, the PXE boot `grub.cfg` file, `grub/grub.cfg` in this example, provides menu options for booting further. The PXE boot `grub.cfg` config file specifies the locations of the kernel and `initrd` files relative to the tftp directory.

- Configure the HTTP Directory:

  Configure the HTTP file directory and ISO image by placing a copy of the Base OS 7.0.0 ISO in directory `/local/http/base_os_7.0.0/`. In this example, the full path is `/local/http/base_os_7.0.0/base_os_7.0.0.iso`.

- Configure the TFTP Directory By Using the Following Steps:

  Mount the Base OS 7.0.0 ISO. Assume your mount point is `/mnt`:

  ```
  sudo mount -o loop /local/http/base_os_7.0.0/base_os_7.0.0.iso /mnt
  ```

  Copy the kernel and `initrd` from the ISO to the tftp directory:

  ```
  cp /mnt/casper/vmlinuz /local/tftp/baseos/vmlinuz
  cp /mnt/casper/initrd /local/tftp/baseos/initrd
  ```

  Unmount the Base OS 7.0.0 ISO:

  ```
  umount /mnt
  ```

- Download GRUB binaries and Copy it for PXE Booting into Place:

```
cd /tmp
wget -q http://ports.ubuntu.com/ubuntu-ports/dists/jammy/main/uefi/grub2-
↪arm64/current/grubnetaa64.efi.signed
cp grubnetaa64.efi.signed /local/tftp/
```

2.8.2.2 PXE Server Configuration for Recovery Media

In this example, the directory structure on the HTTP and TFTP server looks like this:

```
/local/
   http/
      fastos/
         usb.customer.tar.gz
   tftp/
      fastos/
         vmlinuz
         initrd
      grub/
         grub.cfg
      grubnetaa64.efi.signed
```

Note

The `vmlinuz` and `initrd` files are specified relative to the TFTP root, `/local/tftp/`; and the location of the ISO, `usb.customer.tar.gz`, is relative to the HTTP root, `/local/http/`.

Here, the DHCP and PXE servers are configured to use the above directory structure. The person responsible for deploying the PXE environment should change the directory names and structure to fit their infrastructure.

You can set up the directory structure on your HTTP and TFTP server similarly.

The contents of the /local/tftp/grub/grub.cfg file should look something like this:

```
set default=0
set timeout=5

menuentry "Install DGX Spark FastOS" {
    linux /fastos/vmlinuz nouveau.modeset=0 console=tty0 console=ttyS0,921600
↪sbsa_gwdt.action=1 noui pxeinstall=true fastos_usbimg_url=http://<Server IP>
↪/fastos/usb.customer.tar.gz ip=dhcp static_ip=<static ip>:<SUBNET>[:
↪<gateway ip>:<INTERFACE>]
    initrd /fastos/initrd
}
```

When the system being installed boots via PXE, boot files located on `/local/tftp` are retrieved from the TFTP server. (In this example, the TFTP server is provided by the `xinetd` service whose configuration file, `/etc/xinetd.d/tftp`, specifies that the boot files are located on `/local/tftp`.) When a system is PXE booted, the `grubnetaa64.efi.signedi` file that is designated in the DHCP server's `dhcpd.conf` file is retrieved by TFTP transfer (see *Configure Your DHCP Server*). By default, after the `grubnetaa64.efi.signed` is booted, the PXE boot `grub.cfg` file, `grub/grub.cfg` in this example, provides menu options for booting further. The PXE boot `grub.cfg` config file specifies the locations of the kernel and `initrd` files relative to the tftp directory.

- Configure the HTTP Directory:

  Configure the HTTP file directory and ISO image by placing a copy of the DGX OS exactly as `/local/http/usb.customer.tar.gz`. Download the recovery media archive file (tar.gz format) from https://developer.nvidia.com/downloads/dgx-spark/dgx-spark-recovery-image-1.135.29.tar.gz and change its name to `usb.customer.tar.gz`.

- Configure the TFTP Directory By Using the Following Steps:

  Untar the `usb.customer.tar.gz` file to the `/tmp` directory:

  ```
  tar xpfv /tmp/usb.customer.tar.gz
  ```

  Copy the kernel and `initrd` from the tarball to the tftp directory:

  ```
  cp /tmp/usbimg.customer/usb/vmlinuz /local/tftp/fastos/
  cp /tmp/usbimg.customer/usb/initrd /local/tftp/fastos/
  ```

- Download GRUB binaries and Copy it for PXE Booting into Place:

  ```
  cd /tmp
  wget -q http://ports.ubuntu.com/ubuntu-ports/dists/jammy/main/uefi/grub2-
  ↪arm64/current/grubnetaa64.efi.signed
  cp grubnetaa64.efi.signed /local/tftp/
  ```

2.8.3. TFTP and HTTP Server Verification

After you have set up all elements of your PXE server, prior to doing a PXE install, verify that the TFTP and HTTP servers are working properly.

TFTP Server Verification

To verify that the TFTP server has been set up correctly, from a different system on the same subnet, use `tftp` to get one of the files that will be obtained via `tftp` during the PXE boot. In this example, the TFTP server has been set up to serve files from `/local/tftp`. The grub configuration file, `grub.cfg` is located on `/local/tftp/grub/grub.cfg`; therefore, from the TFTP command prompt, request grub.cfg via `get grub/grub.cfg`.

```
cd /tmp
tftp <TFTP_Server_IP>
get grub/grub.cfg
quit
```

HTTP Server Verification

To verify that the HTTP server has been set up correctly, use the `wget` command to get one of the files that will be obtained via HTTP during the PXE boot. In this example, the HTTP server has been set up to serve files from `/local/http`. The tarball, `usb.customer.tar.gz`, is located on `/local/http/fastos/usb.customer.tar.gz`; therefore, test the HTTP request to retrieve `usb.customer.tar.gz` by running the following commands:

```
cd /tmp
wget http://<HTTP Server IP>/fastos/usb.customer.tar.gz
```

2.8.4. Useful Parameters for Configuring Your System's Network Interfaces

`static_ip=<static ip>:<gateway ip>`: tells the initramfs to configure the system's interfaces using a static IP address.

2.8.5. Parameters Unique to the Spark OS Installer

- `usb.skipfw`: Skips the firmware update step during the installation.
- `usb.shell`: Starts a shell instead of installing the OS.
- `noui`: Disables the user interface during the installation.
- `autoinstall`: Enables the autoinstall feature during the installation.

2.8.6. Configure Your DHCP Server

The DHCP server is responsible for providing the IP address of the TFTP server and the name of the bootloader file in addition to the usual role of providing dynamic IP addresses. The address of the TFTP server is specified in the DHCP configuration file as `next-server`, and the bootloader file is specified as `filename`. The architecture option can be used to detect the architecture of the client system and to serve the correct version of the grub bootloader (x86, IA-32, ARM, and so on).

An example of the PXE portion of `dhcpd.conf` is:

```
class "pxeclients" {
  match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
    filename "grubnetaa64.efi.signed";
    next-server <TFTP_Server_IP>;
    option root-path "/local/tftp";
}
```

For example, this is a dhcpd.conf file:

```
authoritative;
default-lease-time 120;
max-lease-time 120;
ddns-update-style none;


subnet 192.168.99.0 netmask 255.255.255.0 {
pool {
   range 192.168.99.2;

# Required for PXE Boot
class "pxeclients" {
   match if substring (option vendor-class-identifier, 0, 9) = "PXEClient";
      filename "grubnetaa64.efi.signed";
      # TFTP Server IP address
      next-server 192.168.99.1;
      option root-path "/local/tftp/";
```

(continues on next page)

(continued from previous page)

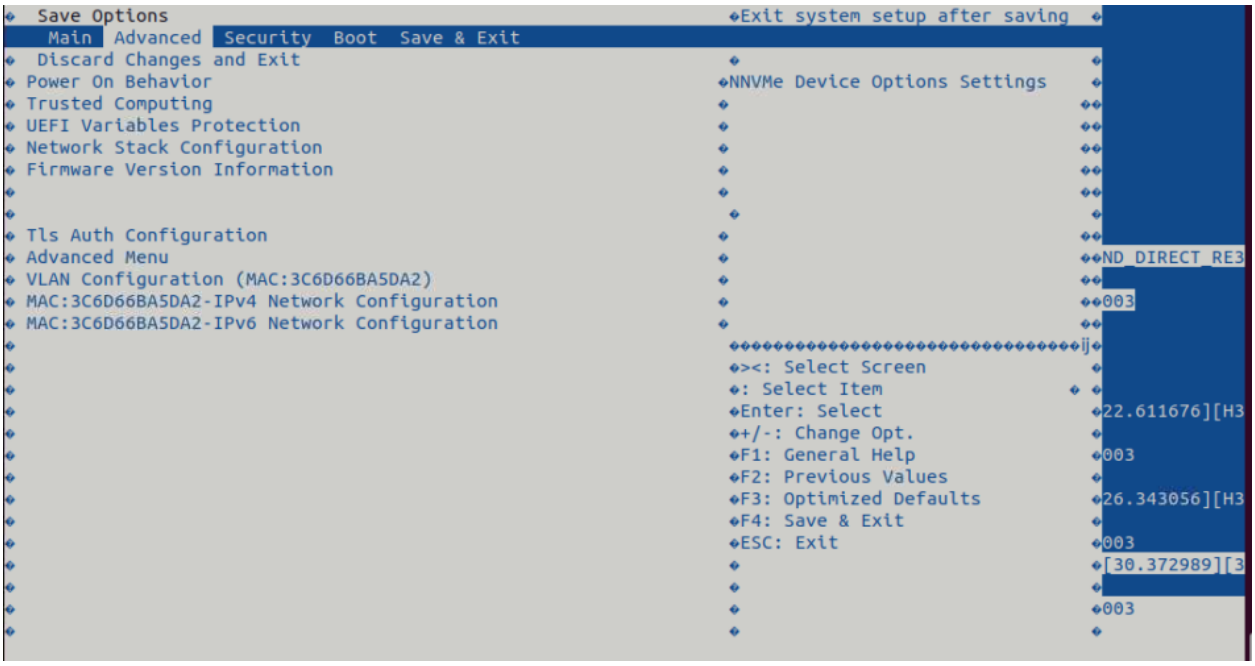
```
}

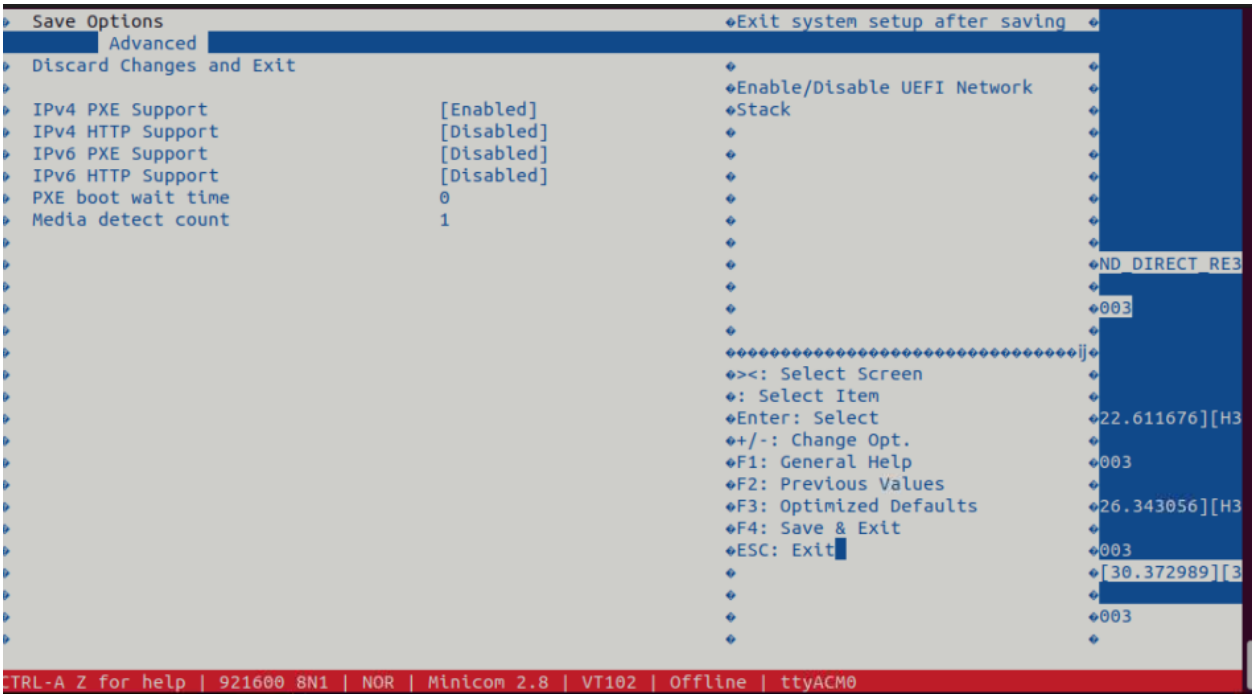
}
}
```

2.8.7. Enable PXE Boot on the DGX Spark

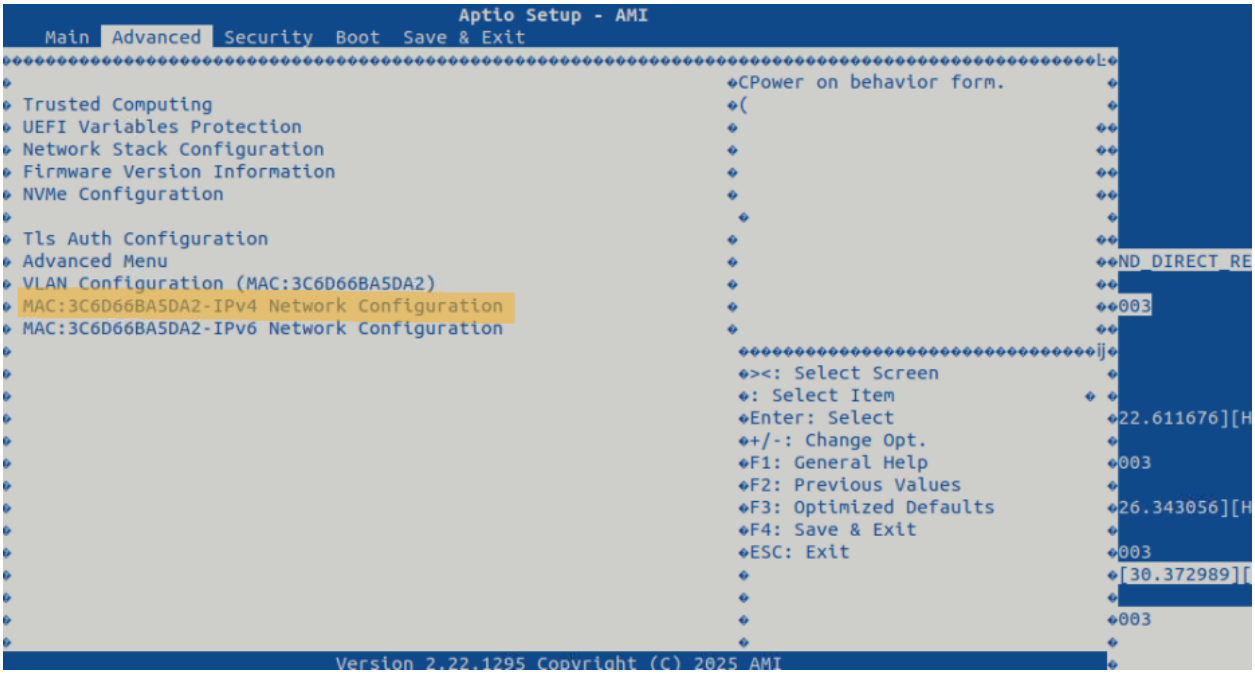
Before enabling PXE Boot, you need to disable the secure boot feature in the UEFI menu or enroll the grubnetaa64.efi.signed in the UEFI menu.

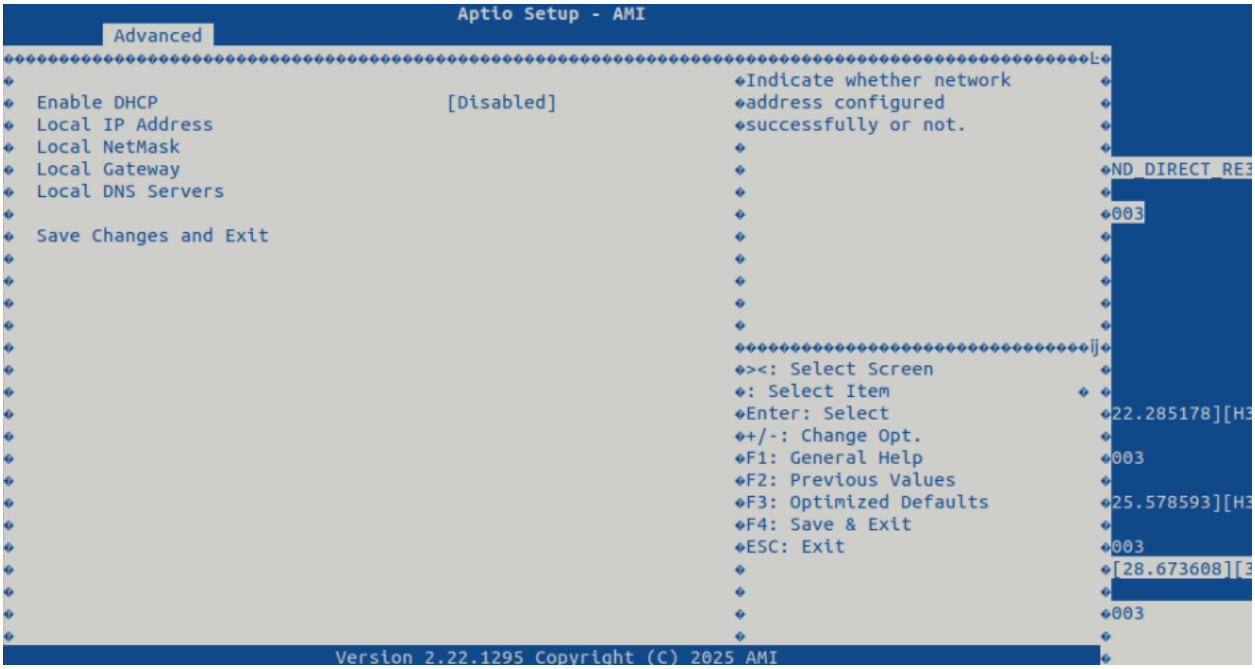
Enable PXE Boot on the DGX Spark by enabling it in `Network Stack Configuration` menu.





Select the NIC you want to use for PXE Boot and enable PXE Boot.





Save and Exit.

After you finished, you can use the UEFI boot menu to do PXE boot and modify the boot order. See https://docs.nvidia.com/dgx/dgx-spark-uefi/boot-tab.html.

2.9. Enterprise Manageability

Enterprise IT teams that operate DGX Spark systems at scale can use the following sections for fleet lifecycle integration and for custom installation when removable media or on-premises mirrors replace direct use of public package sources.

> **Note**
>
> **Enterprise lifecycle integration scripts.** Procedures in *Enterprise Lifecycle Integration* reference a downloadable collection of example Python tools and shell scripts that are not preinstalled on DGX Spark systems. Use them as reference material and adapt them within your own enterprise management platforms. Download Enterprise Lifecycle Integration Scripts package (ZIP) and extract the archive on your integration host, then follow `README.md` in the extracted tree. Optional `install.sh` scripts are included if you choose to deploy examples on endpoints.

2.9.1. Enterprise Lifecycle Integration

2.9.1.1 How to Use This Information

This section provides information on the anticipated uses and limitations of this information.

2.9.1.1.1 Intended Audience

This information is written for the following audience:

- Enterprise IT administrators and endpoint operations teams
- Platform engineering or SRE teams operating fleets at scale
- Security operations teams that require evidence-driven workflows
- Support and escalation teams that need standardized diagnostics artifacts

2.9.1.1.2 Document Scope

The following are within the scope of this document.

- Lifecycle-driven operational playbooks. For example, the following workflow:

Procure Provision Monitor Maintain Respond Retire

- Agentless remote execution model (SSH) with a strict stdout JSON contract
- Evidence minimization and artifact handling patterns
- Wrapper patterns for enterprise platforms (for example, job systems, orchestration, and configuration management)

The following are not within the scope of this document.

- Prescriptive identity architecture (for example, IAM design, key management, and PKI)
- Organization-specific network segmentation and firewall policy
- Business or legal policy decisions (such as retention durations and legal hold workflows)
- Detailed application-layer configuration beyond manageability requirements
- Detailed ISO repack, USB `OEMDATA` layout, and cloud-init seed procedures (see *Custom Installation with cloud-init*)

2.9.1.1.3 Document Conventions

This document uses the following conventions:

- Collectors are read-only tools safe to run frequently and at high concurrency.
- Controllers change state and must be gated by changing windows, rings or waves, and validation.
- Tools return exactly one bounded JSON document on stdout per invocation.
- Large outputs and deep evidence are stored as artifacts; stdout returns pointers only.

When you see inline code, it represents literal strings, such as file names, flags, fields, or paths.

2.9.1.1.4 Operational Guardrails

Note: Read this before running controllers.

The following are guardrails to consider.

- Run controllers only in approved change windows with staged rollouts (for example pilot waves broad).
- Keep stdout JSON bounded to avoid platform truncation.

- Treat artifact creation as on-demand; store and retain artifacts per policy.
- Separate transport or auth failures (platform layer) from tool failures and endpoint health failures.

2.9.1.2 Support Boundaries and Operational Responsibilities

This section clarifies what is provided and supported as part of the DGX Spark manageability baseline versus what remains the responsibility of the enterprise IT environment. It is intended to reduce ambiguity during deployment, escalation, and long-term operations.

2.9.1.2.1 Supported Baseline Posture

2.9.1.2.1.1 Supported Baseline (high-level)

DGX Spark enterprise manageability is designed and validated against the supported baseline OS image (DGX OS).

Baseline Topic	Guidance
OS baseline	DGX OS is the supported baseline for the guide's operational contract.
Driver stack	Driver behavior, versions, and compatible combinations are validated relative to the baseline.
Firmware alignment	Firmware inventory and update posture are interpreted relative to the baseline and platform capabilities.
Tooling	Production tools installed into DGX_spark_management/bin/ are intended for fleet automation.
Reference scripts	Landscape scripts are reference implementations and may require adaptation for production governance.

2.9.1.2.1.2 Shipped Software Baseline and SBOM Source

For DGX Spark Founders Edition, customers who require a software inventory or SBOM can generate one from the DGX Spark System Recovery Image. The recovery image represents the recovery media to reset to factory settings for the latest software image.

For DGX Spark systems from other manufacturers, use the recovery image or software delivery process provided by the respective OEM. OEM variant images might differ from the DGX Spark Founders Edition recovery baseline.

For the recovery media download and recovery procedure, refer to *System Recovery*. For current component version information, refer to *DGX Spark Release Notes*.

2.9.1.2.2 Management Platform Support Posture

Canonical Landscape is the primary recommended management platform for DGX Spark, and its license is included as part of the Ubuntu Pro entitlement. Landscape provides user management, policy deployment, and OS, firmware, and driver updates from a local repository. For enrollment steps, see *Ubuntu Pro and Landscape client enrollment*.

In addition, ecosystem tools such as Ansible, Puppet, Tanium, and others may offer ARM64-capable agents that can be used for broader fleet management, including approval workflows and rollback

strategies. These alternatives are supported solely by their respective vendors. Any integration examples referenced in this document do not imply NVIDIA support or validation.

2.9.1.2.3 Reimaging Posture

Reimaging may be required in certain customer workflows, but it has consequences for predictability and supportability.

2.9.1.2.3.1 Reimaging Posture

Reimaging away from the supported baseline OS can change the behavior across drivers and firmware.

Scenario	Operational Expectation
Baseline DGX OS	Predictable behavior: Spark Manageability Guide playbooks apply directly.
Baseline and approved updates	Predictable within validated update channels; confirm before and after evidence.
Non-baseline OS image	Higher variance; validate collectors and controllers; adjust wrappers and evidence paths as needed.
Mixed fleet baselines	Increase drift and support complexity; maintain baseline identity records per ring or group.

2.9.1.2.4 Division of Responsibility (NVIDIA vs Enterprise IT)

NVIDIA Provides (Typical)	Enterprise IT Owns (Typical)
DGX OS baseline image and guidance	Identity strategy, authentication, RBAC, and secrets management
Reference tools for inventory, diagnostics and update control	Orchestration platform selection, job packaging, scheduling, and rings and waves
Reference scripts demonstrating platform patterns (Landscape)	Change management, approvals, and maintenance windows
Evidence paths and recommended minimization model	Artifact storage, retention policies, legal hold, and ticket linkage
Escalation guidance and support interfaces	Network segmentation, firewall rules, and access pathways
Controller Class	Minimum Governance Expectation
Update control (spark_updatectl.py)	Change window with ring rollout and precheck and postcheck evidence. Rollback awareness is retained.
Diagnostics bundles (spark_diagctl.py bundle modes)	On-demand only. Artifacts are handled through evidence store. Ticket linkage is recommended.

2.9.1.3 DGX Spark Overview, Enterprise Challenges, Lifecycle Backbone, and the JSON/Agentless SSH

DGX Spark is being introduced into environments that already have mature endpoint operations. This section establishes a lifecycle-following framework and a simple operational contract so enterprise tooling can manage DGX Spark consistently at fleet scale.

Key Takeaways

- Treat DGX Spark as a enterprise endpoint with an appliance mindset.
- Standardize on SSH for remote execution and JSON on stdout as the integration contract.
- Use artifacts only when deep evidence is required; keep routine results small and bounded.

2.9.1.3.1 Capabilities Summary

This matrix summarizes the current DGX Spark manageability capabilities exposed by this repository. It is intended to help an Enterprise IT administrator quickly understand the following:

- What exists today and how it is delivered (installed tool vs reference script)
- What evidence is produced (stdout JSON vs artifacts).

How to Read This Table

- **Production (installed)**: Fleet-ready tooling installed into DGX_spark_management/bin/.
- **Reference (in-place)**: Example scripts designed for Canonical Landscape constraints; not installed by default.
- **Evidence model**: stdout JSON is the primary contract; artifacts are generated only when deep evidence is required.

Table 5: DGX Spark manageability capabilities (summary)

Capability area	Integration overview	Delivery	Primary commands	Evidence produced
Identity and asset acceptance	Capture stable identifiers and an "as-received" acceptance snapshot.	Production	`device_identity.py`, `os_build_identity.py`	stdout JSON; optional on-device JSON record
Hardware configuration inventory	Enumerate key hardware configuration (CPU/GPU/SSD/NIC/memor	Production	`hardware_config.py`	stdout JSON; optional on-device JSON record
Firmware inventory	Report firmware versions (UEFI/BIOS, NIC, SSD, GPU as available).	Production	`firmware_reporte.py`	stdout JSON; optional on-device JSON record
Driver inventory	Report key driver versions (GPU/NIC/storage/USB as implemented).	Production	`driver_inventory.py`	stdout JSON; optional on-device JSON record
Software inventory	Enumerate installed software inventories for drift and compliance contexts.	Production	`software_inventor.py`	stdout JSON; optional on-device JSON record
UEFI-backed tags (optional)	Read/write UEFI-backed metadata tags for fleet workflows (if enabled).	Production	`NVAIAread`, `NVAIAwrite`	stdout JSON (read/write result); optional on-device record
Health posture and diagnostics (L1)	Return bounded health posture signals for monitoring and triage.	Production	`spark_diagctl.py`	stdout JSON summary
Deep evidence bundles (L2)	Generate targeted or full diagnostics bundles when escalation requires evidence.	Production	`spark_diagctl.py`	stdout JSON + artifact pointer(s)
Reset / reboot context	Explain reboot/reset reasons for stability correlation and incident triage.	Production	`reset_reason_rep.py`	stdout JSON; optional on-device JSON record
Controlled SW/FW updates	Expose update posture and controlled update operations within maintenance windows.	Production	`spark_updatectl.py`	stdout JSON; optional update evidence artifacts depending on mode
Landscape reference execution	Run platform-friendly checks and store per-run evidence under a run directory.	Reference (`landscap` scripts)	`landscape_*` scripts	Short stdout + on-device evidence under `<runid.log>`

2.9.1.3.2 DGX Spark Overview

DGX Spark is deployed as an enterprise-managed, locally accelerated AI companion device, often alongside standard PC fleets, to provide on-prem or edge AI compute where latency, privacy, data gravity, or disconnected operations make cloud-only workflows impractical.

In enterprise terms, DGX Spark behaves less like a general-purpose end-user PC and more like a managed endpoint appliance:

- It runs NVIDIA's base OS (DGX OS).
- It is typically administered remotely, at scale, with minimal local interaction.
- It should integrate into existing IT operations: inventory, monitoring, patching, incident response, and retirement.

Documentation Principle: Treat DGX Spark as a first-class enterprise endpoint but manage it with an appliance mindset: baseline control, automation, and evidence-driven operations.

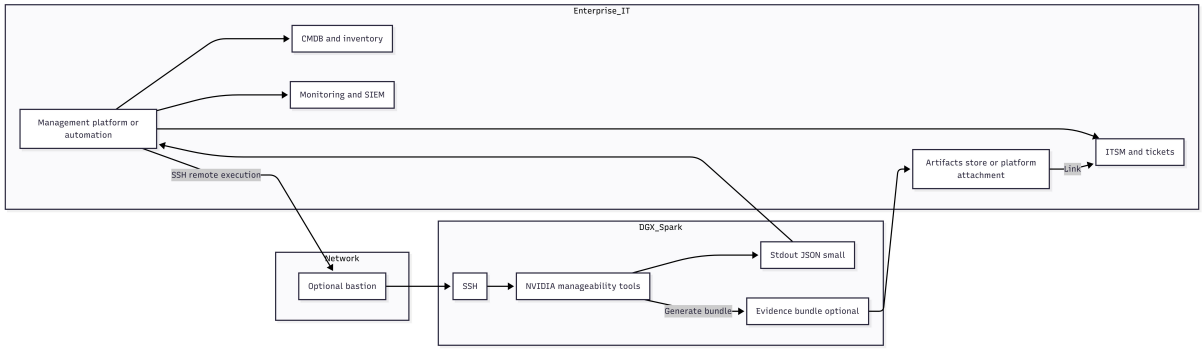


Figure 1: DGX Spark context (PC fleet + AI workloads + enterprise tooling).

2.9.1.3.3 Enterprise Management Challenges

Enterprise IT teams managing PCs globally typically rely on:

- Rich device identity and hardware inventory
- Standardized patching and reboot orchestration
- Predictable driver and firmware servicing channels
- Mature remote diagnostics and support bundles
- CMDB integration, drift detection, and reporting. Linux-based devices often break those assumptions:
- Management approaches differ across distros and package managers
- Endpoint agents vary by platform and may not be desirable in regulated environments
- Ad-hoc scripting without consistent outputs leads to fragile automations
- Troubleshooting evidence collection can be inconsistent, large, and noisy. DGX Spark's manageability approach addresses these challenges by providing:
  - Bounded, machine-ingestible outputs (JSON)
  - An agentless control-plane model (SSH)
  - Optional artifact bundles when deep evidence is required
  - Lifecycle-aligned playbooks that map to standard IT operations

PC-Centric Expectation	DGX Spark Operational Equivalent
Identity and inventory are always queryable	Standard SSH collectors emit bounded JSON snapshots for ingestion.
Patching and reboots are orchestrated	Maintenance-window playbooks: precheck update and reboot postcheck, all returning JSON.
Driver and firmware servicing is predictable	Standardize on a supported baseline (DGX OS) and validate drift through inventory JSON.
Support bundles are standardized	Generate evidence artifacts only when needed and reference them from the JSON result.
Drift detection and reporting are continuous	Record baseline OS build, firmware, drivers, and critical packages. Compare them over time.

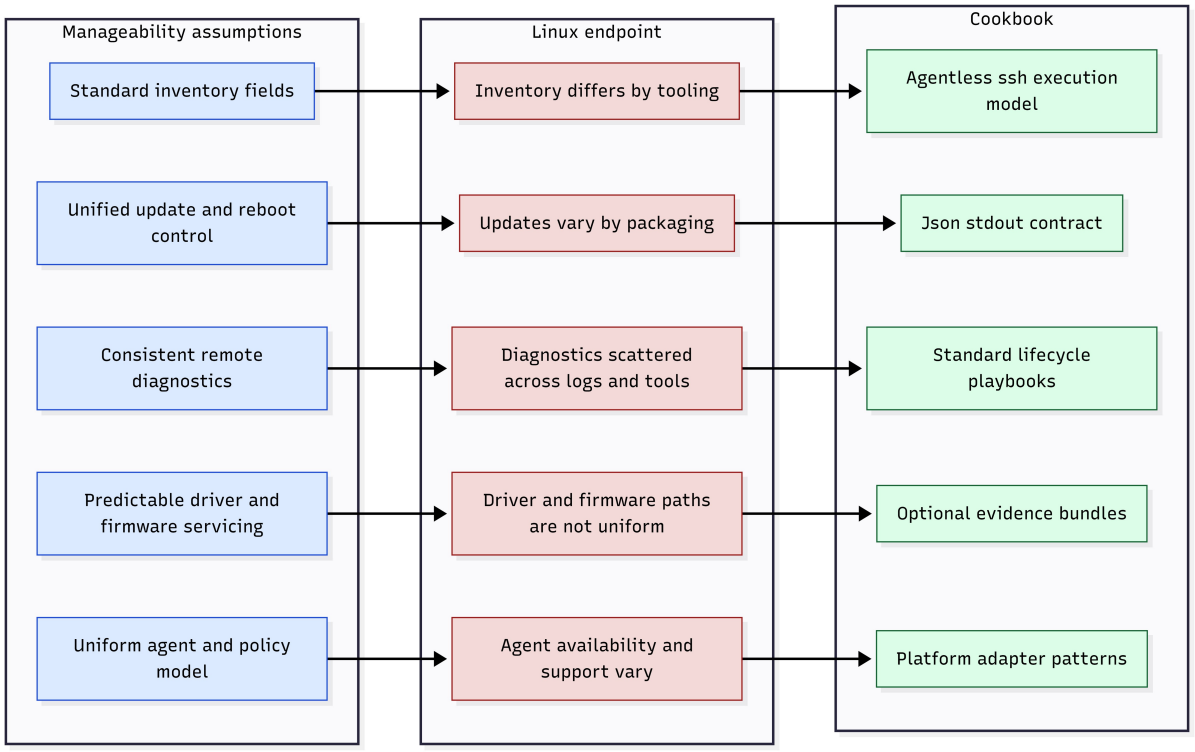


Figure 2: Manageability assumptions vs Linux endpoint realities.

2.9.1.3.4 Lifecycle Backbone (How Enterprise IT Actually Runs Fleets)

The information in this section is organized around a lifecycle backbone that enterprise IT already understands. It deliberately starts before the device is powered on, because procurement decisions (SKUs, support, warranty, spares, standards alignment, and so on) drive long-term operational cost.

Procurement and Receiving

- Define standard SKUs and configurations, support entitlements, and regional logistics.
- Establish asset identity expectations (serial and UUID conventions, labeling, CMDB records).
- Plan sparing, RMA flows, and replacement pools aligned to global sites.

Initial Provisioning

- Establish device identity, hardware, firmware, software inventory, and baseline state.
- Apply initial configuration required for remote management (SSH reachability, time sync, and so on).
- Record enrollment metadata and ownership tags (as defined by your IT processes).
- Apply automated first-boot configuration with cloud-init and repacked BaseOS media where required. See *Cloud-init for DGX Spark* and *Custom Installation with cloud-init*.

Ongoing Monitoring

- Run health checks and alerting signals.
- Detect drift from known good baselines (such as OS build, firmware set, driver set, critical packages).
- Analyze reset reasons and stability indicators to catch systemic issues early.

Maintenance Windows

- Coordinate controlled updates and reboots within change windows.
- Validate update outcomes and preserve rollback safety.
- Enforce staged rollouts (rings and waves) to reduce fleet risk.

Incident Response

- Collect targeted evidence or full diagnostics bundles when needed.
- Package artifacts for escalation (such as, engineering, NVIDIA support, or OEM workflows).
- Execute remediation consistently and record outcomes for postmortems.

End-of-Life

- Execute factory reset aligned to retirement policy.
- Produce retirement evidence (such as method, timestamps, success and failure, and artifact references).
- Handle redeploy, transfer, and disposal with chain-of-custody documentation.

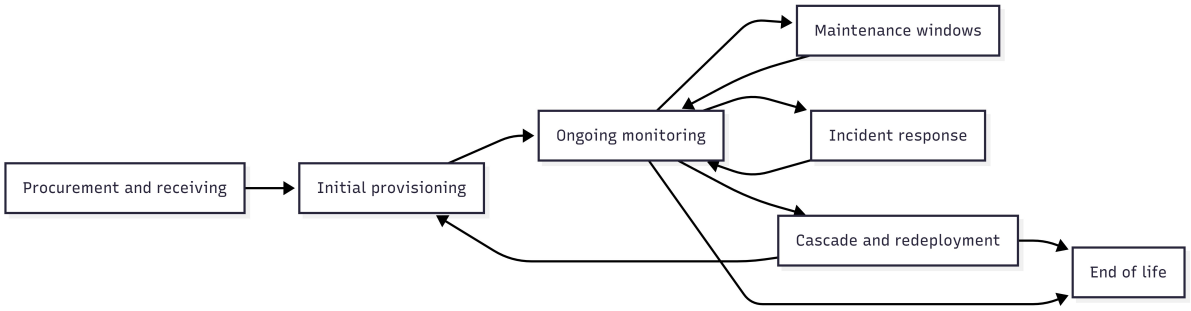


Figure 3: Lifecycle backbone (Procure Provision Monitor Maintain Respond Retire).

2.9.1.3.5 JSON-First, Agentless SSH Execution Model

The information in this section standardizes on one universal remote control plane:

- **Remote execution:** SSH (works from Windows and from enterprise management platforms)
- **Output contract:** JSON on stdout (small, bounded; or machine-ingestible)
- **Deep evidence:** Artifacts (tarballs and log bundles) referenced by JSON and pulled only when needed. This model intentionally avoids requiring a resident management agent on the DGX Spark endpoint.

Existing platforms orchestrate SSH execution and ingest JSON results.

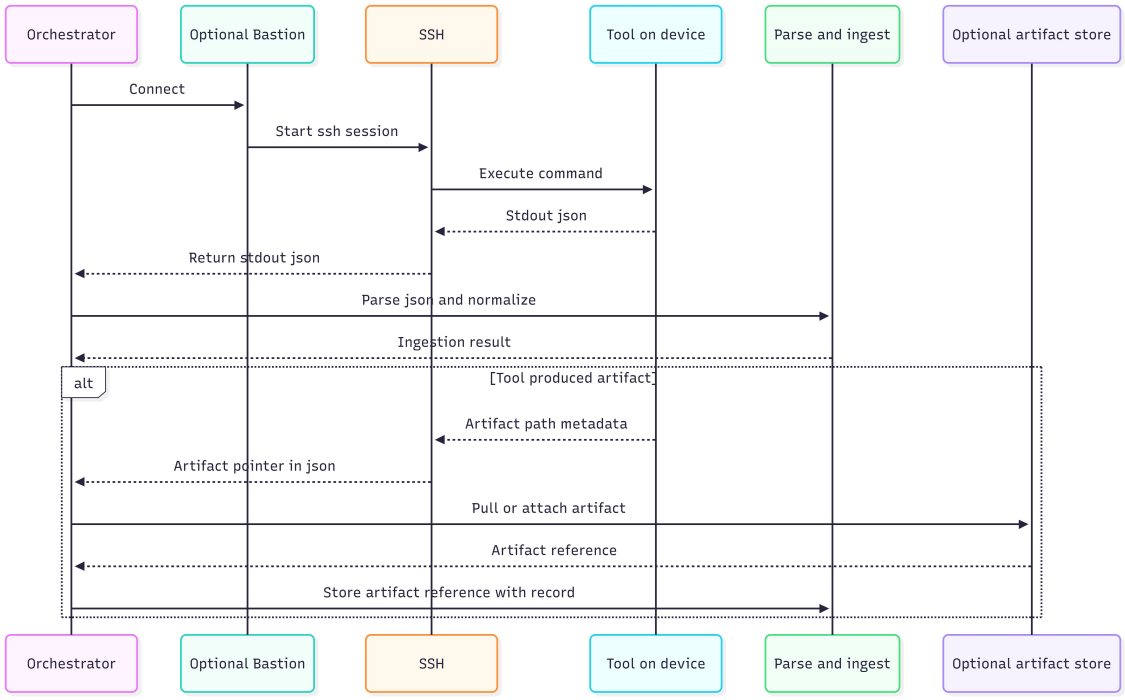


Figure 4: Orchestrator SSH Tool stdout JSON parse or ingest (plus optional artifact pull).

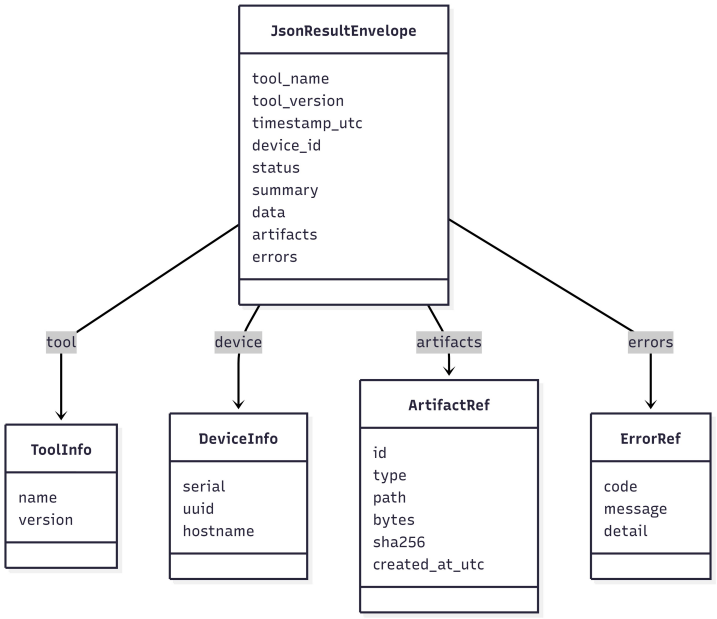


Figure 5: JSON result envelope (fields) and artifact pointer model.

2.9.1.3.6 Simple Operations

IT teams can run the entire control plane from endpoints and servers using built-in OpenSSH capabilities:

- Direct SSH command execution
- Scripting and scheduling
- JSON capture and forwarding into CMDB, SIEM, and monitoring pipelines

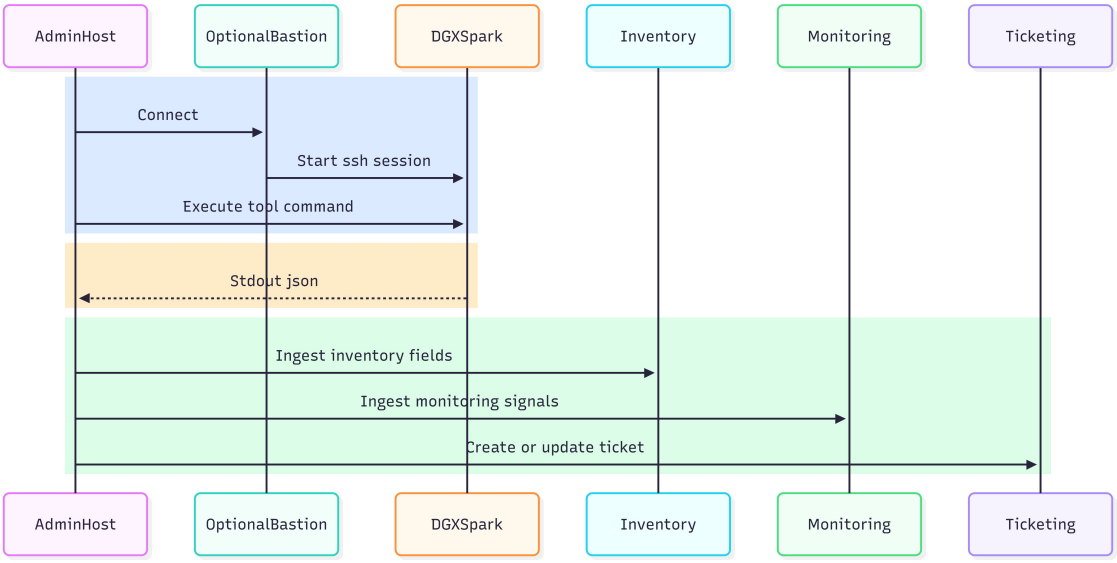


Figure 6: Admin workstation or server SSH DGX Spark JSON ingestion targets.

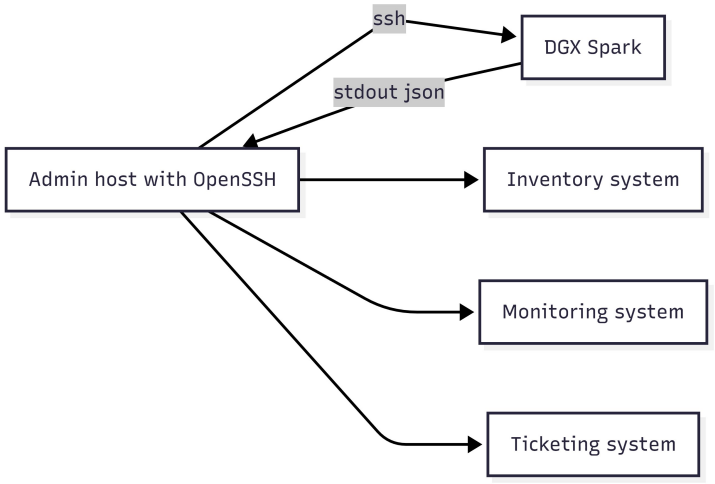


Figure 7: Admin host SSH invocation example flow

2.9.1.3.7 Artifact Strategy

Use two tiers of operational evidence:

- **stdout JSON:** Bounded, predictable, and easy to store and index
- **artifacts:** Collected selectively (incident response and escalation), stored with retention controls; and CMDB stores pointers only

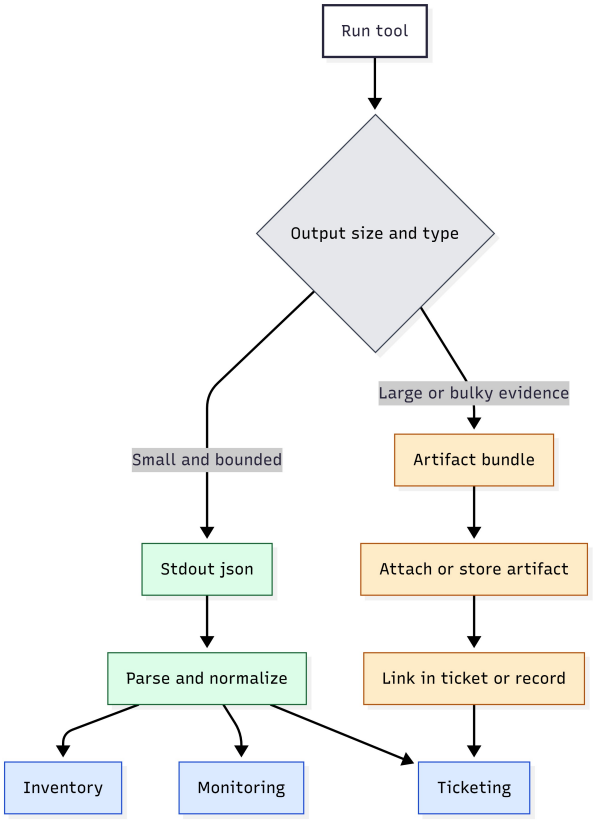


Figure 8: Decision split—small JSON vs large artifacts (pointers + retention).

Artifact Type	Typical Retention	Notes
Routine health snapshots	30–90 days	Prefer JSON-only; avoid artifacts unless needed.
Incident diagnostics bundles	90–180 days	Retain for root cause analysis and escalation cycles.
Security audit evidence	180–365+ days	Align with compliance and legal hold policy.
Retirement proofs	Per policy	Retention period set by chain-of-custody or enterprise governance.

2.9.1.3.8 Operational UX Details

This subsection describes the practical operator experience, such as what to run, what you get back, and how to handle evidence in a way that scales in enterprise job systems.

2.9.1.3.8.1 Operator Workflow in One Screen

- Pick the **lifecycle intent** (such as inventory, drift check, health posture, update window, incident evidence, and retire).
- Run the corresponding **installed command** over SSH.
- Capture **stdout JSON** as the authoritative record.
- Retrieve **artifacts** only when JSON reports they exist or escalation requires deeper evidence.

2.9.1.3.8.2 Quick start: The four most common actions

Intent	Command to Run	What to Store
Acceptance snapshot (as-received)	device_identity.py os_build_identity.py	stdout JSON for CMDB; optional on-device JSON record
Baseline inventories (provisioning)	hardware_config.py firmware_reporter.py driver_i nventory_reporter.py software_i nventory_reporter.py	stdout JSON for drift anchors; retain build + driver + firmware fields
Monitoring (routine posture)	spark_diagctl.py rese t_reason_reporter.py	stdout JSON for health/stability signals
Incident escalation (deep evidence)	spark_diagctl.py (bundle modes)	stdout JSON + artifact pointer(s); retrieve bundle only when needed

2.9.1.3.8.3 Stdout JSON is the API (contract)

Stdout JSON is the integration contract. Recommended capture: stdout (verbatim), rc, stderr, ts, host. See *JSON result capture (stdout as the API)* for the standard envelope fields.

2.9.1.3.8.4 Evidence Handling: Summary First, Artifacts on Demand

Apply the rule in *Artifact Strategy*. If the result is small, keep it in stdout JSON. If the result is large (such as logs or bundles), generate or retrieve it as an artifact. Store artifacts in your normal evidence store and link them back to the run record or ticket.

Scenario	Operational Handling
Routine monitoring	stdout JSON only; run frequently; bounded outputs
Drift investigation	stdout JSON summaries; artifact only if a deep diff is required
Incident response L1	stdout JSON + minimal targeted log excerpts
Incident response L2	stdout JSON + artifact bundle; retrieve and attach to ticket
Retirement evidence	stdout JSON “certificate” + optional artifact evidence per policy

2.9.1.3.8.5 Common Failure Classes

When automation fails at scale, the fastest recovery comes from classifying failures consistently:

Failure Class	What it Usually Means	First Action
Transport/auth	SSH could not connect or authenticate; device unreachable; key or policy issue.	Fix connectivity or credentials in the orchestration layer; rerun a collector.
Privilege	Command requires sudo or access to privileged system interfaces.	Grant least-privilege sudo for the tool; re-run; validate no interactive prompts.
Tool execution	Missing dependency, unexpected OS state, or tool internal error.	Check stderr + tool log directory; compare against supported baseline.
Endpoint degraded	Tool ran successfully but returned warning or failure status due to device health signals.	Collect targeted diagnostics; escalate to L2 bundle if needed.

2.9.1.4 Integration Patterns: SSH Execution from Management Platforms

This section describes how enterprise platforms execute remote actions over SSH, capture results, and feed them into downstream systems. The goal is portability. The same operational pattern should work whether you trigger jobs from Windows, a jump host, or a central orchestration tier.

Key takeaways: See *JSON-first, agentless SSH execution model* and *Artifact Strategy* for the execution and evidence model. Design for fleet realities: output limits, timeouts, concurrency, and predictable privilege boundaries.

2.9.1.4.1 The Universal Remote Execution Model

At scale, enterprise tooling converges on a consistent execution loop:

1. Select targets (static groups, rings, or dynamic inventory queries).
2. Execute a remote command over SSH (often through a job, package, or policy).
3. Capture stdout, stderr, and exit code.
4. Parse stdout JSON into normalized fields.
5. Ingest results into CMDB, monitoring, and ITSM workflows.
6. (Optional) Retrieve artifacts when JSON indicates they exist.

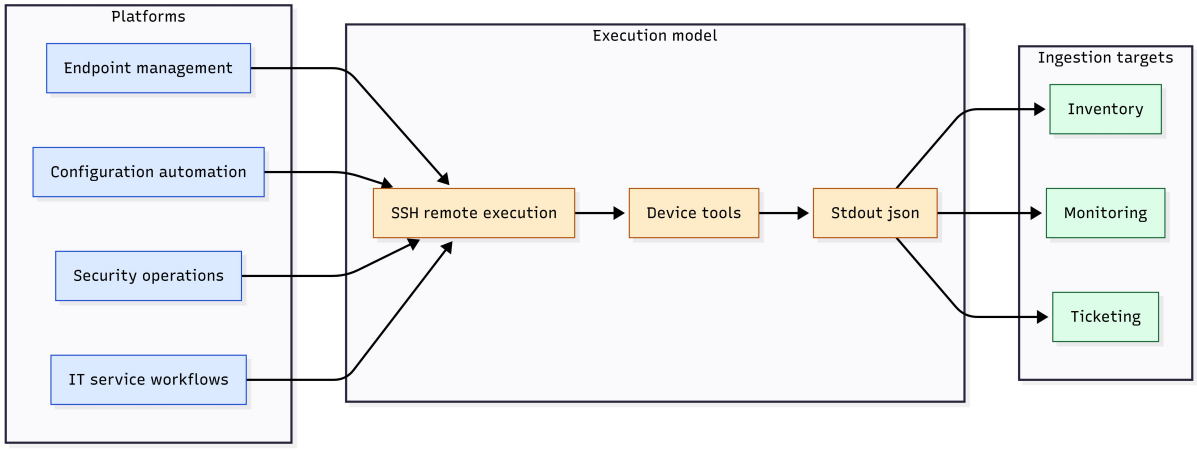


Figure 9: Universal model — platform SSH device tool stdout JSON ingest; optional artifact pull.

2.9.1.4.2 SSH Execution

Many enterprise IT teams prefer a simple operational model.

- Admin devices includes an OpenSSH client for interactive use and automation.
- PowerShell provides a natural surface for fan-out, scheduling, and JSON capture.
- Output should be stored verbatim and parsed off-box.

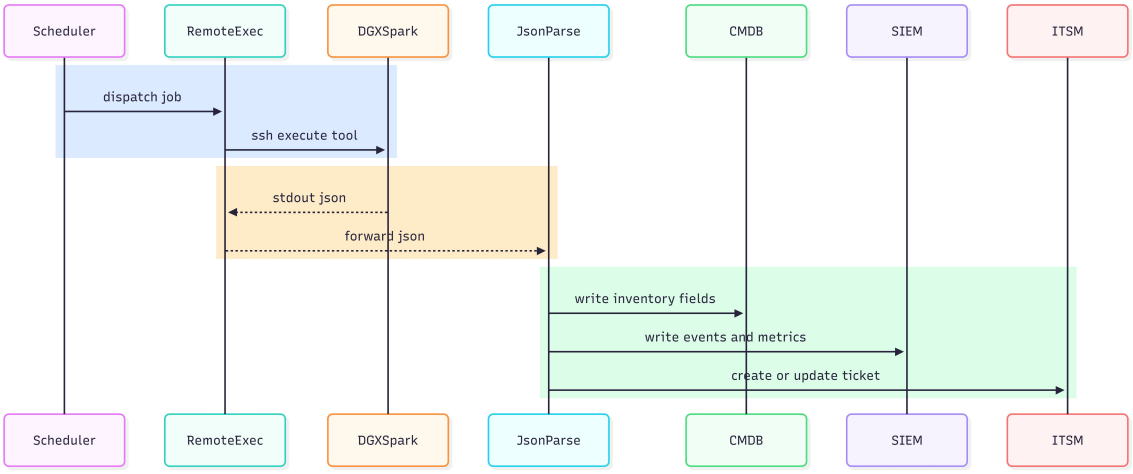


Figure 10: SSH invocation capturing stdout JSON to file.

ssh -o BatchMode=yes -o StrictHostKeyChecking=accept-new nvidia@DGX_HOST “sudo /usr/local/bin/dgx-mgmt/spark_diagctl.py” | Out-File -Encoding utf8 .\ spark_diagctl.py $doc = Get-Content .\ spark_diagctl.py -Raw | ConvertFrom-Json

$doc.status

2.9.1.4.3 SSH Access Patterns for Enterprise Operations (Non-Prescriptive)

The information in this section is intentionally non-prescriptive about identity and IAM. In practice, most enterprises converge on one of these connectivity patterns:

- Direct SSH per device (common in labs and smaller networks).
- Bastion or jump-host mediated SSH (typical for segmented networks).
- Brokered SSH execution (platform runs jobs from a central execution tier).

Operational considerations that matter at fleet scale:

- Stable addressing or an authoritative inventory source
- Non-interactive authentication and execution
- Predictable sudo permissions for tools
- Separation between read-only collectors and state-changing controllers

Pattern	Strengths	Tradeoffs
Direct per-device SSH	Simple and transparent; minimal infrastructure	Harder at scale; network exposure; credential sprawl risk
Bastion or jump host	Fits segmentation; centralizes access control and logging	Capacity planning; potential bottleneck
Brokered execution	Central scheduling, targeting, and reporting; easier fleet automation	Depends on platform constraints; adds abstraction

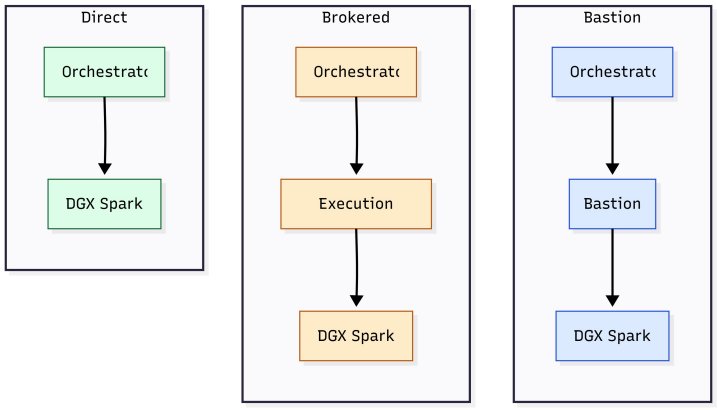


Figure 11: SSH connectivity patterns — direct vs bastion vs brokered execution.

2.9.1.4.4 JSON Result Capture (stdout as the API)

(Model described in *JSON-first, agentless SSH execution model* and *Artifact Strategy*) To keep integrations uniform across platforms, it is assumed that:

- Orchestration wrappers store stdout verbatim (or as close as possible)
- Parsing happens off-box (in the platform, not on the device)

Operational requirements:

- stdout JSON is the authoritative contract
- stderr is retained for debugging, but is not the primary data plane
- If a tool fails before emitting JSON, the wrapper should emit a standard failure envelope

Field	Purpose
tool	Stable identifier of the command or tool invoked
ts	UTC timestamp for correlation
host	Target identity used by the orchestrator (hostname or asset ID)
status	ok
rc	Exit code from the tool execution
duration_ms	Runtime to support SLOs and debugging
summary	Small bounded summary suitable for indexing
warnings	Optional list of non-fatal issues
artifacts	Optional list of artifact pointers (ok, data, errors, meta)

{

“tool”: “spark_diagctl.py “,

“ts”: “2026-01-12T21:17:00Z”,

“host”: “DGX_HOST”,

“status”: “ok”,

“rc”: 0,

“duration_ms”: 842,

“summary”: {

“disk”: “ok”,

“network”: “ok”, “drivers”: “ok”

},

“warnings”: [], “artifacts”: []

}

2.9.1.4.5 Artifact Retrieval Patterns

Artifacts are used when:

- A health signal indicates anomaly
- Incident response requires evidence
- Escalation requires a standardized support bundle
- Audits require retained evidence beyond JSON summaries

Common patterns:

- Pull model: Orchestrator retrieves artifact through scp/sftp after job completion.
- Store-and-link model: Orchestrator uploads artifact to internal object storage and stores a link in the job record or ticket.
- Retention model: Artifacts are retained per policy and then deleted automatically.

Artifact Type	Typical Trigger	Notes
Diagnostics bundle	Incident or escalation	Prefer store-and-link; include hashes; keep sizes predictable
Focused logs	Single failing signal	Smaller than full bundles; faster retrieval and triage
Configuration snapshot	Drift investigation	Pair with a baseline record for comparisons
Update evidence	Maintenance window validation	Attach to change ticket when required

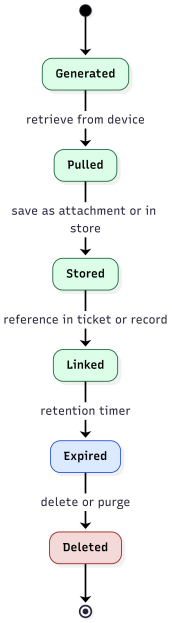


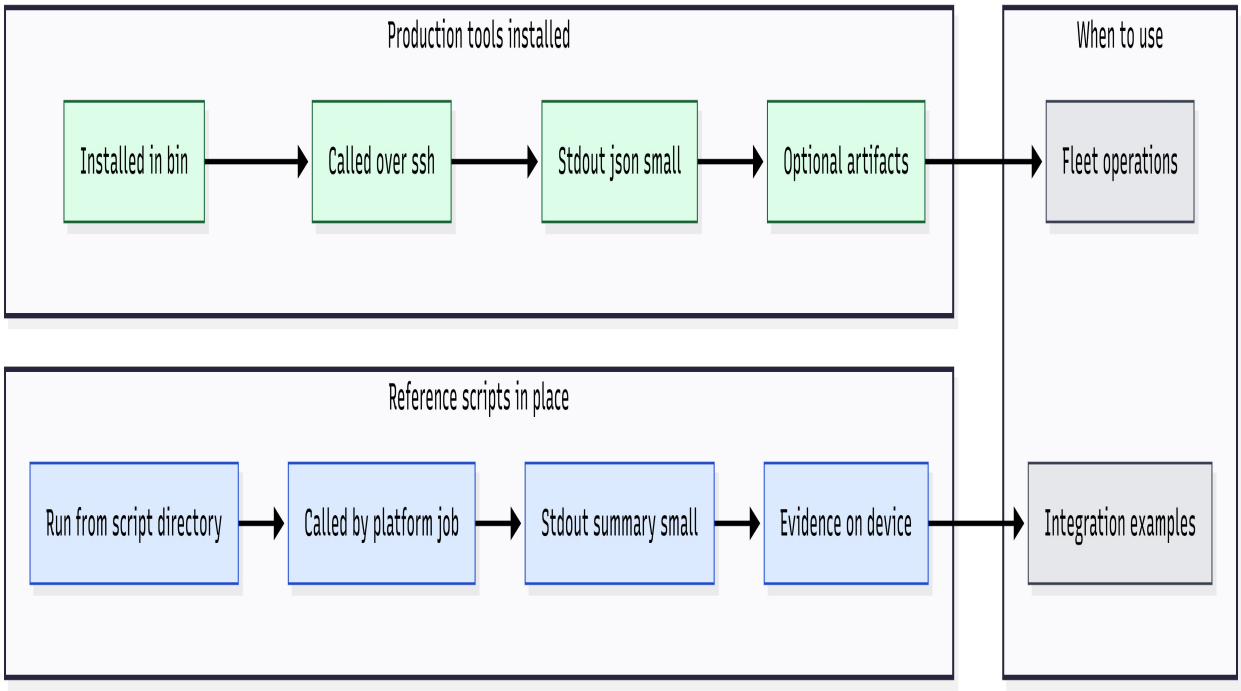
Figure 12: Artifact lifecycle — generate retrieve store link expire.

2.9.1.5 Implementation

This section is meant to help an Enterprise IT admin (or integrator) quickly answer:

- What tooling does NVIDIA provide?
- Where is the source code and documentation for each tool?
- What gets installed on-device and what runs in-place as reference code?
- Where do outputs and logs land on the DGX Spark device?
- Which tools or scripts are relevant to each lifecycle stage

For top-level directory layout and command-to-source-path mapping, see *Repository layout reference*.



2.9.1.5.1 Repository Content

Download Enterprise Lifecycle Integration Scripts package (ZIP) and extract it on your integration host.

The package is a collection of example Python tools and shell scripts intended to help enterprise IT teams manage DGX Spark systems at scale. The examples use agentless, SSH-based remote execution with structured JSON output for integration into your enterprise management platforms. NVIDIA provides these materials for reference and customer adaptation. They are not a supported production software offering. The examples illustrate capabilities across the device lifecycle, from procurement and provisioning through monitoring, maintenance, incident response, and retirement, with no resident agent on the device. Capabilities include the following:

- Hardware and software inventory
- Firmware and driver reporting
- Health diagnostics
- Controlled updates
- Security posture checks

The archive includes tool and script sources, *reference shell scripts*, optional per-tool `install.sh` scripts, and companion documentation referenced in this section.

This repository contains two distinct implementation types:

2.9.1.5.1.1 Production Tools (11)

These are production-ready, stdlib-only Python tools designed for fleet automation with:

- JSON-first output
- Configuration defaults with CLI override support
- Safety guardrails for state-changing operations
- Comprehensive documentation and test coverage

The following is the installed command location on the device:

- DGX_spark_management/bin/

The following are the production tools functional areas and source roots:

- clear_asset_information/ (7 tools)
- controlled_sw_fw_updates/ (1 tool)
- remote_ops_remediation/ (2 tools)
- data_protection_privacy/ (1 tool)

Each production tool follows a consistent structure:

- {functional_area}/{tool_name}/src/
- {functional_area}/{tool_name}/config/
- {functional_area}/{tool_name}/install.sh
- Installed entry point(s) land in DGX_spark_management/bin/

2.9.1.5.1.2 Canonical Landscape Reference Scripts (8)

These are reference implementations designed for Canonical Landscape “remote script execution” constraints. The device must be enrolled in Landscape first; see *Ubuntu Pro and Landscape client enrollment*.

The *reference shell scripts* and their per-script `README.md` files are included in Enterprise Lifecycle Integration Scripts package (ZIP). After you extract the archive, follow `LANDSCAPE_REFERENCE_SCRIPTS_SETUP.md` at the archive root or `docs/core_docs/01_core_overview/03_LANDSCAPE_REFERENCE_SCRIPTS_SETUP.md` before using those scripts.

- Minimal error handling (examples, not production frameworks)
- stdout output intended to be short and platform-friendly
- Detailed evidence stored locally on the device per run

Reference Scripts Run In-Place

- They are not installed into bin/ by default.
- They live under their functional area folders with landscape_ prefix naming.

Reference Script Functional Areas

- attestable_conformance_regulatory/ (1 script)
- resilience_recovery_rollback/ (4 scripts)
- network_enterprise_connectivity/ (2 scripts)
- security_posture_vuln_response/ (1 script)

Reference script structure:

- {functional_area}/landscape_{script_name}/README.md
- {functional_area}/landscape_{script_name}/{script_name}.sh

Script Path (Repo)	Primary Script	Purpose	Stdout + Evidence
attestable_conformance_regulatc landscape_signing_verification/	sign-ing_verificatio	APT signing verification	Short stdout; detailed evidence stored per run on device
resilience_recovery_rollback/ land-scape_verified_boot_integrity/	veri-fied_boot_inte	Verified boot integrity reference	Short stdout; per-run evidence under run directory
resilience_recovery_rollback/ land-scape_recovery_backup_levels/	recov-ery_backup_le	Recovery/backup level reference	Short stdout; per-run evidence under run directory
resilience_recovery_rollback/ land-scape_factory_reset_reprovision/	fac-tory_reset_rep	Factory reset + reprovision	Short stdout; per-run evidence under run directory
resilience_recovery_rollback/ landscape_health_watchdogs/	health_watchd	Health watchdog reference	Short stdout; per-run evidence under run directory
network_enterprise_connectivity/ landscape_collect_package/	col-lect_package.s	Support bundle collection reference	Short stdout; evidence bundle stored on device
network_enterprise_connectivity/ landscape_retrieve_logs_stdout/	re-trieve_logs_st	Bounded log retrieval reference	Short stdout; bounded excerpts; detailed evidence on device
secu-rity_posture_vuln_response/ landscape_encryption_at_rest/	encryp-tion_at_rest.sh	Encryption-at-rest reference	Short stdout; evidence stored per run on device

2.9.1.5.1.3 When to Use Reference Scripts

Use reference scripts when management platform constraints are the dominant design factor (for example, strict stdout limits or “run script remotely” paradigms). For broad enterprise automation, use production tools installed in `bin/`.

2.9.1.5.2 Primary Entry Points

After you download and extract the script package (see *Repository Content*), use the following files in the extracted tree for the authoritative directory layout, reading order, and per-tool references. Paths below are relative to the root of the extracted script package.

- `docs/core_docs/00_INDEX.md` — Documentation index and reading order
- `docs/core_docs/01_core_overview/01_PROJECT_README.md` — Overview, quick start, naming conventions
- `docs/core_docs/01_core_overview/02_PROJECT_STRUCTURE.md` — Directory tree and runtime layout
- `docs/core_docs/01_core_overview/03_LANDSCAPE_REFERENCE_SCRIPTS_SETUP.md` — Landscape reference script framework

I want to...	Read this first
Get a quick start view	docs/core_docs/01_core_overview/01_PROJECT_README.md
Understand folder structure and runtime layout	docs/core_docs/01_core_overview/02_PROJECT_STRUCTURE.md
Integrate Landscape reference scripts	docs/core_docs/01_core_overview/03_LANDSCAPE_REFERENCE_SCRIPTS_

2.9.1.5.3 Tool Deep-Dive Documentation

Each production tool has the following:

- A tool folder README at {functional_area}/{tool_name}/README.md
- Centralized “production tools” documentation referenced by `docs/core_docs/00_INDEX.md`. *Tool locations and reference code map* points to these documents per tool and summarizes the following:
- Command synopsis and options
- JSON output structure and key fields
- Troubleshooting and known limitations
- Integration notes

2.9.1.5.4 Installed Commands on the DGX Spark

This subsection provides information about the installed commands on the DGX Spark.

2.9.1.5.4.1 Installed Command Directory

When deployed, production tools are installed here:

- DGX_spark_management/bin/ Expected installed commands:
- bin/device_identity.py
- bin/hardware_config.py
- bin/firmware_reporter.py

- bin/os_build_identity.py
- bin/driver_inventory_reporter.py
- bin/software_inventory_reporter.py
- bin/NVAIAwrite
- bin/NVAIAread
- bin/spark_updatectl.py
- bin/spark_diagctl.py
- bin/reset_reason_reporter.py

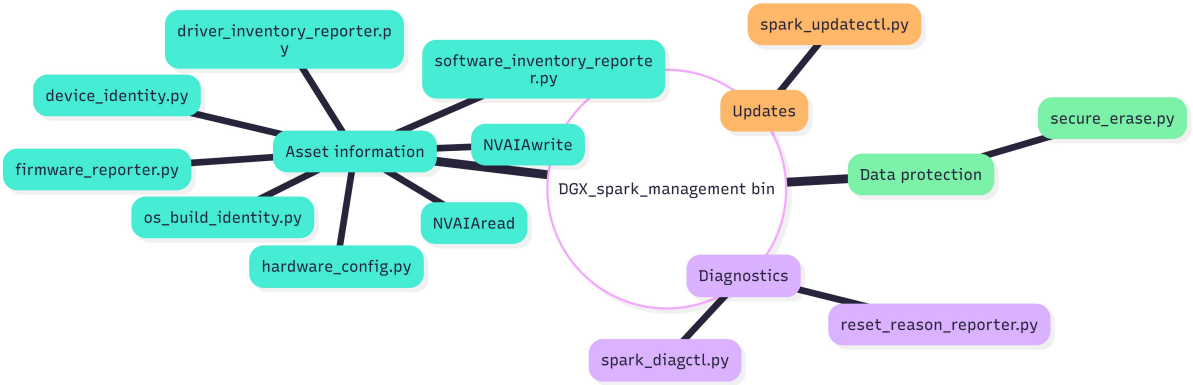


Figure 15: Example listing of DGX_spark_management/bin/ on a device.

2.9.1.5.5 Runtime Outputs and Logs

Production tools and reference scripts write runtime output to:

- /var/lib/dgx_spark_management/{functional_area}/{tool_or_script_name}/ Examples:
- /var/lib/dgx_spark_management/clear_asset_information/hardware_inventory_collector/ device_identity.json
- /var/lib/dgx_spark_management/controlled_sw_fw_updates/update_control_plane/ status.json
- /var/lib/dgx_spark_management/remote_ops_remediation/ diagnostic_collector/diagnostics_full.json

Production logs are found under:

- /var/log/dgx_spark/{functional_area}/{tool_name}/

Examples:

- /var/log/dgx_spark/clear_asset_information/ hardware_inventory_collector/device_identity.log
- /var/log/dgx_spark/controlled_sw_fw_updates/update_control_plane/spark_updatectl.log
- /var/log/dgx_spark/remote_ops_remediation/diagnostic_collector/spark_diagctl.log

Landscape reference scripts store per-run evidence under:

- /var/lib/dgx_spark_management/{functional_area}/{landscape_script}/run_<UTC>/

Category	Location Pattern
Production runtime outputs	/var/lib/dgx_spark_management/ {functional_area}/{tool_name}/
Production logs	/var/log/dgx_spark/ {functional_area}/{tool_name}/

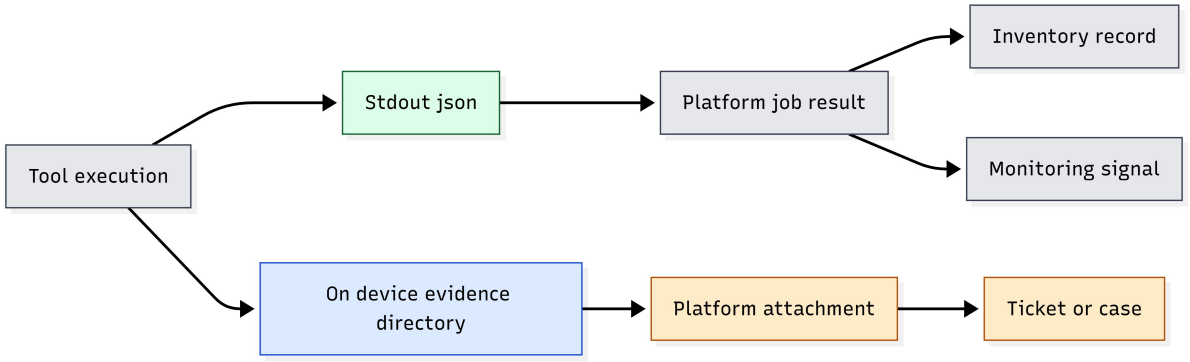


Figure 16: Evidence flow — stdout JSON vs on-device evidence directory vs platform attachment.

2.9.1.5.6 Repository Layout Reference

Use this subsection to locate folders and source files in the `DGX_spark_management/` script package. For the full directory tree and runtime layout, see `docs/core_docs/01_core_overview/02_PROJECT_STRUCTURE.md` in the extracted archive.

2.9.1.5.6.1 Top-Level Repository Layout

Path	Purpose
bin/	Installed production tool entrypoints (what operators run on end-points)
docs/	Architecture, structure, deployment, and tool references
clear_asset_information/	Production tools: identity, hardware, firmware, OS build, drivers, software, asset tags
controlled_sw_fw_updates/	Production tool: update control plane (reboot coordination, rollback reporting)
remote_ops_remediation/	Production tools: diagnostics collector, reset reason reporter
data_protection_privacy/	Production tool: data protection and privacy
at-testable_conformance_regulat	Landscape script: APT signing verification
resilience_recovery_rollback/	Landscape scripts: boot integrity, backup levels, factory reset, watchdogs
net-work_enterprise_connectivity/	Landscape scripts: support bundle collection, log retrieval
secu-rity_posture_vuln_response/	Landscape script: encryption-at-rest reporting
enrollment_identity_access/	Planned functional area placeholder
integration_automation/	Planned functional area placeholder
lifecycle_business_ops/	Planned functional area placeholder
packaging/	Packaging artifacts (for example, systemd units)
tools/	Development tooling helpers

Where operators spend time

- Operators and integrators typically use `bin/` (installed commands) and `docs/` (usage and integration guidance).
- Developers work primarily under functional area folders.

2.9.1.5.6.2 Production Tools: Command-to-Source Mapping

Each production tool folder includes `README.md`, `src/`, `config/`, and `install.sh` (installs into `bin/`).

2.9.1.6 Clear Asset Information (Seven Tools)

Installed command	Source root	Notes
device_identity.py	clear_asset_information/hardware_inventory_c	Stable identifier through SMBIOS/DMI logic
hardware_config.py	clear_asset_information/hardware_inventory_c	CPU/GPU/SSD/NIC/memory enumeration
firmware_reporter.py	clear_asset_information/firmware_version_repc	BIOS/UEFI/NIC/SSD/GPU firmware enumeration
os_build_identity.py	clear_asset_information/os_build_identity_repo	OS build + DGX identity
driver_inventory_repc	clear_asset_information/driver_inventory_repor	GPU/NIC/storage/USB drivers
soft-ware_inventory_repo	clear_asset_information/software_inventory_re	dpkg/snap/pip/docker enu-meration
NVAIAwrite / NVA-IAread	clear_asset_information/asset_tag_manager/sr	UEFI-backed metadata store (read/write)

2.9.1.7 Controlled Software/Firmware Updates (One Tool)

Installed command	Source root
spark_updatectl.py	controlled_sw_fw_updates/update_control_plane/src/spark_updatectl.py

2.9.1.8 Remote Ops and Remediation (Two Tools)

Installed command	Source root
spark_diagctl.py	remote_ops_remediation/diagnostic_collector/src/spark_diagctl.py
reset_reason_reporter.py	remote_ops_remediation/reset_reason_reporter/src/reset_reason_reporter.py

2.9.1.8.1 Lifecycle Mapping to the Actual Code in This Repository

This section maps the lifecycle found in *Lifecycle backbone (how enterprise IT actually runs fleets)* to the tools and scripts in this repository.

2.9.1.8.1.1 Procurement and Receiving

Relevant production tools:

- bin/device_identity.py
- bin/os_build_identity.py
- bin/hardware_config.py

- bin/firmware_reporter.py

Stage Objective	Recommended Tools	Outputs to Capture
Acceptance snapshot	Identity + build + HW + FW	stdout JSON + stored on-device JSON paths

2.9.1.8.1.2 Initial Provisioning

Relevant production tools:

- Identity, hardware, firmware, OS build, driver inventory, and software inventory
- optional: bin/NVAIAwrite and bin/NVAIAread if UEFI-backed tags are used

Baseline Set	Primary Tools	Drift Anchor Fields
Identity + build + inventories	device_identity.py os_build_identity.py hardware_config.py firmware_reporter.py driver_i nventory_reporter.py software_i nventory_reporter.py	OS build identity; firmware set; driver set; and critical package inventory hashes and counts

Optional first-boot automation uses cloud-init with repacked BaseOS media; see *Cloud-init for DGX Spark*.

2.9.1.8.1.3 Cloud-init for DGX Spark

This section explains cloud-init basics and how it supports automated first-boot provisioning of DGX Spark in enterprise deployments.

2.9.1.8.2 What is cloud-init

Cloud-init is a widely used initialization framework that runs during the early boot process of Linux systems. It allows system configuration and provisioning tasks to be automated at first boot using simple configuration files. Originally developed for cloud environments, cloud-init is equally effective for provisioning physical devices such as DGX Spark systems.

For DGX Spark deployments, cloud-init enables administrators to apply configuration policies automatically when a device is first powered on, eliminating the need for manual setup.

Typical uses include:

- Setting the system hostname and identity
- Creating administrator accounts and installing SSH keys
- Configuring networking or Wi-Fi profiles
- Installing packages or applying update policies
- Installing corporate certificates or proxy settings
- Registering the system with enterprise management platforms

Cloud-init executes during the early stages of the boot process and normally runs once per instance, making it well suited for automated provisioning workflows.

2.9.1.8.3 DGX Spark Procedures

On DGX Spark, cloud-init implements first-boot provisioning during the **Initial Provisioning** lifecycle stage. Site-specific configuration (users, packages, out-of-box experience behavior, and optional local package or firmware mirrors) is delivered through repacked BaseOS media, OEM cloud-init seeds, and optional `OEMDATA` USB layout.

For step-by-step procedures, USB partitioning, repack scripts, example `user-data` files, and verification workflows, see *Custom Installation with cloud-init*.

For general cloud-init module reference and behavior, see https://docs.cloud-init.io/en/latest/.

2.9.1.8.3.1 Ongoing Monitoring

The following are relevant production tools:

- bin/spark_diagctl.py (health posture and targeted collection modes)
- bin/reset_reason_reporter.py
- (Optional) Periodic drift re-checks using os_build_identity, driver_inventory, and firmware_reporter

Signal	Source Tool	Artifacts (if any)
Health posture	spark_diagctl.py	Optional bundle modes
Reset context	reset_reason_reporter.py	None

2.9.1.8.3.2 Maintenance Windows

The following are relevant production tools:

- bin/spark_updatectl.py
- Validation tools: os_build_identity, spark_diagctl, optional driver or firmware tools

Phase	Tools	Evidence
Pre-check	Build identity + health	stdout JSON
Update control	spark_updatectl.py	stdout JSON + optional artifact
Post-check	Build identity + health	stdout JSON

2.9.1.8.3.3 Incident Response

Relevant production tools:

- Level 1: Targeted triage: spark_diagctl health, reset_reason_reporter, plus identity, build, and drivers as needed
- Level 2 Deep evidence: spark_diagctl bundle collection modes. Relevant reference scripts (Landscape environments) include log retrieval and support bundle collection scripts under network_enterprise_connectivity/

Level	Tools and Scripts	Output Handling Expectation
Level 1	Bounded collectors	stdout JSON only
Level 2	Bundle modes / reference scripts	stdout JSON + artifacts/evidence

2.9.1.8.3.4 Cascade and Redeployment

Cascade and redeployment is a re-provisioning motion:

- The device changes user, function, or environment
- The lifecycle loop returns to provisioning steps while preserving linkage to prior history, if desired.

Relevant production tools:

- Provisioning baseline set (identity, build, hardware, firmware, drivers, and software)
- Optional tag tools, if used to mark new function or owner. Relevant reference scripts (Landscape environments) include factory reset and re-provisioning examples under resilience_recovery_rollback/

Motion	Recommended Steps	Evidence to Capture
Re-provision	Baseline re-capture or optional reset	New baseline JSON plus optional reset evidence

2.9.1.9 Tool Locations and Reference Code Map

This section explains the tools and example implementations inside the project repository. The goal is to make it easy to locate the correct scripts, understand what they do, and embed them into enterprise platform wrappers without copying large files.

2.9.1.9.1 Repository Layout Conventions Used in This Section

It is assumed that the repository organizes content into:

- **Tools**: Scripts or binaries invoked remotely (collectors and controllers).
- **Reference code**: Examples used to demonstrate platform integration patterns.
- **Platform wrappers**: Example job scripts for specific management platforms.
- **Artifacts**: Optional bundles (diagnostics or logs) created on demand.

To avoid repeating paths everywhere, define a single canonical base path and reuse it.

Note: In the provided examples, the tool directory is referenced as a single variable (for example, dgx_tool_dir). Replace it with your repo's actual path on the device.

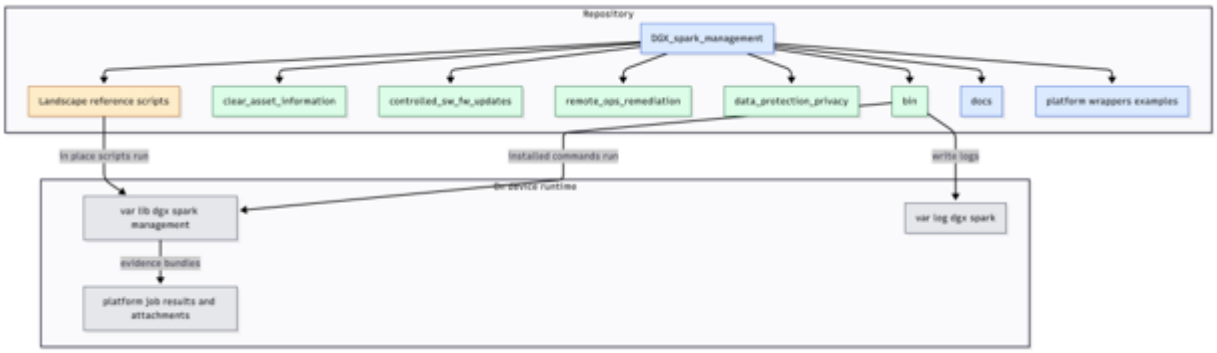


Figure 17: Repo layout — tools, reference code, wrappers, and artifacts.

2.9.1.9.2 Canonical Tool Directory and Naming Scheme

For readability and consistency, it is assumed that:

- Collectors are named *.py and emit one JSON envelope on stdout
- Controllers are also *.json but are gated and validated
- Artifact generators indicate artifact creation and return pointers in artifacts

Category	Convention
Collector	Name ends in .json; read-only; safe to run frequently; bounded output
Controller	Name ends in .json; changes state; gated; includes precheck= and postcheck guidance
Artifact generator	Name indicates bundle creation; stdout JSON returns artifact pointer list

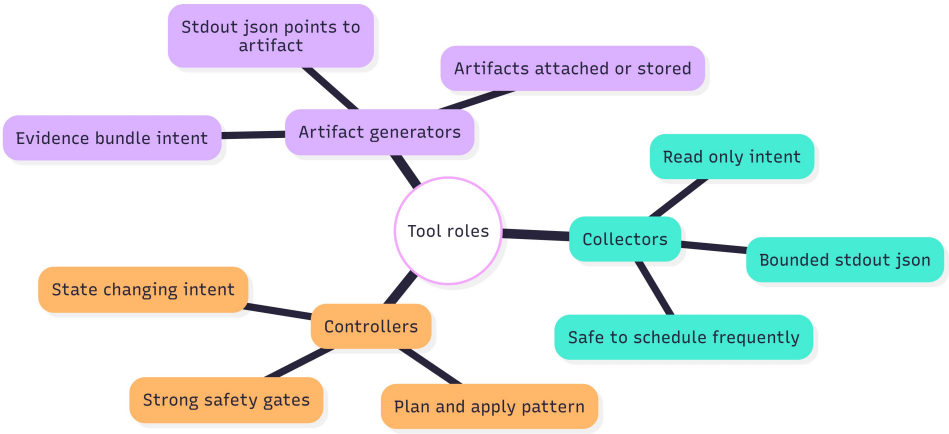


Figure 18: Naming scheme — collectors vs controllers vs artifact generators.

2.9.1.9.3 Reference Code Map

See *Canonical Landscape reference scripts (8)* for the full list of Landscape reference scripts and their purposes. Enrollment is required before running them; see *Ubuntu Pro and Landscape client enrollment*.

2.9.1.9.3.1 How to Use These References

Treat these directories as examples of platform wrapper integration and evidence minimization. For production automation, use the guide's standardized collector and controller tools and keep wrapper scripts thin.

2.9.1.9.3.2 Tool Invocation Patterns

Across all platforms, it is assumed that the same invocation pattern is used:

- SSH executes a tool from the canonical tool directory.
- The tool emits one JSON envelope on stdout.
- The platform stores stdout JSON verbatim and parses it off-box.
- Artifacts are created only on demand; stdout returns pointers.

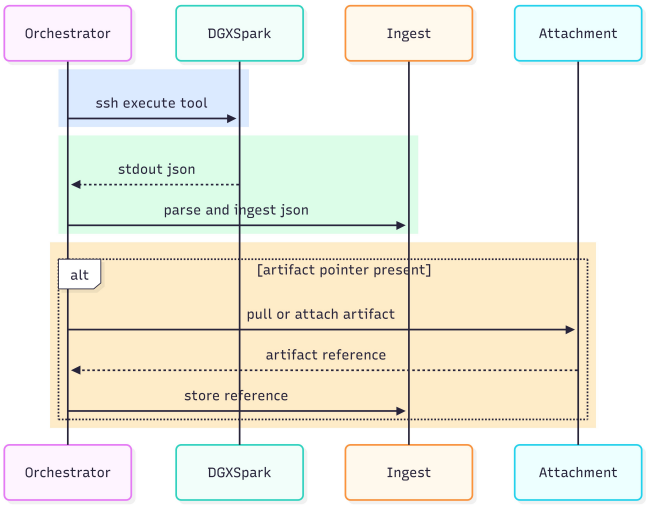


Figure 19: Invocation contract — SSH executes tool stdout JSON ingest; optional artifacts by pointer.

2.9.1.9.3.3 Embedding Paths in Examples

To keep the document readable, examples should not hardcode long paths repeatedly. Use one variable name consistently, and document where the variable is set per platform.

Platform Wrapper	How to Represent the Tool Path
Windows scripts	Set a single variable and call tools via that variable
Ansible	Use a vars: entry such as dgx_tool_dir and reference it in tasks
Landscape jobs	Use the job definition to call a script in-place, or call a canonical tool directory
BigFix/Tanium	Store the SSH wrapper centrally and pass the tool path as a parameter

Recommendation

In later sections, whenever you see dgx_tool_dir, replace it with the actual installed location on the endpoint. Keep the replacement consistent across all playbooks and wrappers.

2.9.1.9.3.4 Evidence Minimization (stdout vs artifacts)

This repository includes examples that write evidence to disk (for later retrieval). That pattern is correct for large evidence and diagnostics, but stdout must remain bounded.

Use this decision rule:

- If the result is small and can be summarized, return it in stdout JSON.
- If the result is large, generate an artifact and return a pointer in stdout JSON.

Case	Preferred Handling
Routine inventory and health signals	stdout JSON summary only
Long logs or large inventories	artifact bundle + pointer in stdout JSON
Incident response deep evidence	artifact bundle + hashes + retrieval hint
Audit or retirement proofs	stdout JSON certificate + artifact evidence as required

For additional information on evidence minimization, see *Artifact Strategy* and *Evidence Handling: Summary First, Artifacts on Demand*.

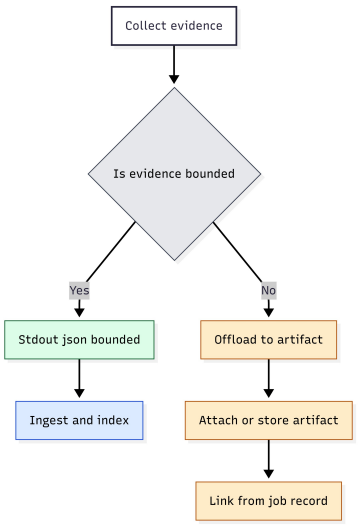


Figure 20: Evidence minimization — bounded stdout JSON vs artifact offload.

2.9.1.10 Platform Wrapper Patterns and Packaging Examples

This section shows examples of how to package the guide's SSH + stdout-JSON tooling model inside common enterprise platforms.

Key takeaways: Consider the same model as found in *JSON-first, agentless SSH execution model* and *Artifact Strategy* (execute capture stdout JSON optionally pull artifacts). Use collectors (safe, frequent) vs controllers (gated, validated).

2.9.1.10.1 Common Packaging Principles (Applies to All Platforms)

2.9.1.10.1.1 Output Sizing

Consider the same principle as in *Artifact Strategy*: keep stdout bounded and use artifacts for large evidence. Most platforms impose stdout or result-size limits. Packaging should treat:

- stdout JSON as the authoritative small record
- Large bundles (diagnostics tarball, large logs, or full inventories) as attachments or artifacts

Output Type	Packaging Guidance
stdout JSON	Bounded summary, stable envelope, safe to index and ingest
stderr	Keep for debugging. Do not rely on it for structured data.
Artifacts	Use for large evidence. Attach or store and return pointers in JSON.

Recommendation

If a platform truncates outputs, treat the truncation as a correctness failure. Reduce stdout size or switch the payload to an artifact generator that returns only a pointer on stdout.

2.9.1.10.2 Ansible Playbook Examples (Agentless SSH-Native)

Ansible aligns naturally with this information because it is SSH-native and can fetch files. These examples focus on:

- Running the installed tools on each host
- Capturing JSON results per host
- Optionally fetching artifacts

Inventory Assumption	Conceptual Guidance
Host grouping	Use groups for rings or waves (such as pilot, wave1, wave2, or broad)
Connection	SSH keys and non-interactive execution; bastion if required
Privilege	Use become for controllers; keep collectors unprivileged when possible
Outputs	Write stdout JSON per-host; collect centrally; parse off-box

2.9.1.10.2.1 Example: Provisioning Baseline (Conceptual)

— name: Provisioning baseline (collectors) hosts: dgx_spark gather_facts: no vars:

dgx_tool_dir: /path/to/tools # set per Part 4 tasks: - name: Collect identity/build

shell: “sudo {{ dgx_tool_dir }}/identity.json” register: identity_out changed_when: false

- name: Collect hardware/firmware/driver/software inventories shell: “sudo {{ dgx_tool_dir }}/inventory.json” register: inv_out changed_when: false
- name: Write results (stdout JSON) per-host copy:

content: “{{ identity_out.stdout }}n” dest: “./out/{{ inventory_hostname }}_identity.json” - name: Write inventory JSON per-host copy:

content: “{{ inv_out.stdout }}n”

dest: “./out/{{ inventory_hostname }}_inventory.json”

2.9.1.10.2.2 Example: Monitoring Snapshot

— name: Monitoring snapshot (collectors) hosts: dgx_spark gather_facts: no vars:

dgx_tool_dir: /path/to/tools # set per Part 4 tasks: - name: Health check

shell: “sudo {{ dgx_tool_dir }}/spark_diagctl.py “ register: health_out changed_when: false

- name: Reset reason

shell: “sudo {{ dgx_tool_dir }}/reset_reason.json” register: reset_out changed_when: false

- name: Save JSON outputs copy:

content: “{{ health_out.stdout }}n” dest: “./out/{{ inventory_hostname }}_health.json”

- name: Save reset reason JSON copy:

content: “{{ reset_out.stdout }}n”

dest: “./out/{{ inventory_hostname }}_reset_reason.json”

2.9.1.10.2.3 Example: Incident Response L2

— name: Incident response L2 (artifact bundle) hosts: dgx_spark gather_facts: no vars:

dgx_tool_dir: /path/to/tools # set per Part 4 tasks:

- name: Trigger diagnostics bundle

shell: “sudo {{ dgx_tool_dir }}/diag_collect.json” register: diag_out

- name: Save diagnostics JSON copy:

content: “{{ diag_out.stdout }}n”

dest: “./out/{{ inventory_hostname }}_diag_collect.json”

- name: Fetch artifact (conceptual) # Replace with a real fetch that reads the artifact pointer from JSON fetch:

src: “/path/on/device/to/artifact.tar.gz”

dest: “./artifacts/{{ inventory_hostname }}_artifact.tar.gz” flat: yes

2.9.1.10.3 Ubuntu Pro and Landscape Client Enrollment

Complete these steps before using the Landscape reference scripts in *Canonical Landscape reference scripts (8)* or the job examples in *Canonical Landscape script examples*. Download and extract Enterprise Lifecycle Integration Scripts package (ZIP) if you have not already done so.

For the most up-to-date information, see Landscape installation and set-up - Landscape documentation.

Scope: These steps target Landscape SaaS enrollment through Ubuntu Pro (`sudo pro enable landscape` with **Self-hosted server?** set to **No**). Self-hosted Landscape deployments can differ; follow Canonical's documentation for those environments.

Landscape SaaS enrollment requires Ubuntu Pro. The following steps enable Ubuntu Pro services and enroll a DGX Spark as a Landscape client.

1. Register for Ubuntu One.

   Go to Ubuntu Pro and register for an Ubuntu One account.

2. Create a Landscape account.

   1. Open a browser.
   2. Navigate to https://landscape.canonical.com.
   3. Sign in using Ubuntu One.
   4. Create a new organization or join an existing one.
   5. Record your account name (organization name). You need this value during client enrollment.

3. Attach Ubuntu Pro on the DGX Spark.

   Landscape SaaS enrollment requires Ubuntu Pro. On the DGX Spark, check status:

   ```
   sudo pro status
   ```

   If the system is not attached, run:

   ```
   sudo pro attach <YOUR_UBUNTU_PRO_TOKEN>
   ```

   Obtain the token from https://ubuntu.com/pro, then verify attachment:

   ```
   sudo pro status
   ```

   You should see **Attached: yes**.

4. Enable Landscape (interactive mode).

   Run:

   ```
   sudo pro enable landscape
   ```

   When prompted, enter:

   - **Self-hosted server?: No**
   - **Computer title:** for example, `dgx-spark-01`
   - **Account name:** your Landscape organization name

   This process installs, configures, and starts the Landscape client.

5. Approve the machine in the Landscape portal.

1. Return to the Landscape web portal.
2. Navigate to **Pending Machines**.
3. Select the DGX Spark system.
4. Click **Accept**.

The system appears in your managed machines list.

6. Verify client operation.

```
sudo systemctl status landscape-client
```

You should see **Active: active (running)**.

If needed, check the logs:

```
sudo tail -n 50 /var/log/landscape/client.log
```

After enrollment, you can use Landscape to:

- Apply tags (for example, DGX, Spark, AI-node)
- Execute remote scripts
- Schedule updates
- Monitor package and security status
- Create access groups
- Define update and compliance policies

2.9.1.10.4 Canonical Landscape Script Examples

Complete *Ubuntu Pro and Landscape client enrollment* before running these jobs.

Script paths below are relative to the root of the extracted Enterprise Lifecycle Integration Scripts package (ZIP).

Canonical Landscape is included because the repository provides reference scripts that illustrate platform job integration. They are not meant to replace production tools. They demonstrate:

- Integrating health and security checks into Canonical Landscape jobs
- Producing bounded stdout results
- Writing evidence to disk for later retrieval

Reference script locations (from *Tool locations and reference code map*):

- attestable_conformance_regulatory/landscape_signing_verification/
- resilience_recovery_rollback/landscape_verified_boot_integrity/
- resilience_recovery_rollback/landscape_recovery_backup_levels/
- resilience_recovery_rollback/landscape_factory_reset_reprovision/
- resilience_recovery_rollback/landscape_health_watchdogs/
- network_enterprise_connectivity/landscape_collect_package/
- network_enterprise_connectivity/landscape_retrieve_logs_stdout/
- security_posture_vuln_response/landscape_encryption_at_rest/

For example, the following is an APT signing verification (conceptual)

Landscape job (conceptual)

- run signing verification

- emit bounded PASS/FAIL/UNKNOWN on stdout # - write evidence to a directory for retrieval

run_signing_verification emit_stdout_summary write_evidence_dir

2.9.1.10.4.1 Example: Verified Boot Integrity

Landscape job (conceptual)

- check verified boot signals

- emit bounded summary

- store evidence for later retrieval

check_verified_boot emit_stdout_summary write_evidence_dir

2.9.1.10.4.2 Example: Factory Reset with Reprovision

Landscape job (conceptual)

- customer-gated reset workflow # - emit

retirement/reset certificate JSON on stdout # -

store evidence and logs

validate_gate_conditions perform_reset emit_stdout_certificate write_evidence_dir

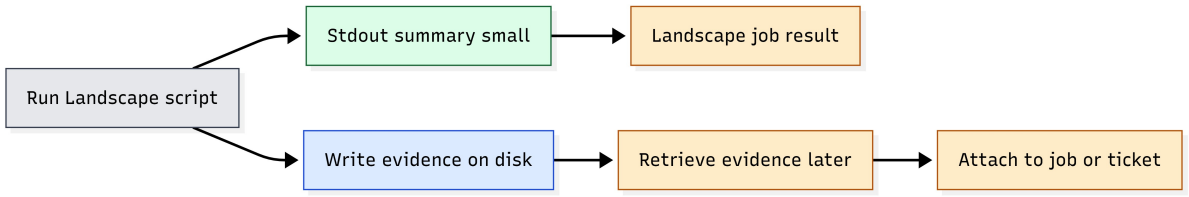


Figure 21: Landscape constraints — stdout summary vs evidence-on-disk retrieval pattern.

2.9.1.10.5 Tanium Package Patterns

Tanium often implements operations through:

- Packages or modules that run scripts
- Results captured in question outputs or package logs
- Attachments handled through platform mechanisms

These patterns show:

- Packaging lifecycle SSH payloads as Tanium packages
- Storing stdout JSON as package output (small)
- Storing large bundles as attachments

Tanium Concept	Documentation Mapping
Package	Wrapper payload that runs SSH tools and captures stdout JSON
Question and result	Ingested JSON summary for reporting and filtering
Package logs	stderr and troubleshooting context
Attachments	Artifacts (bundles) referenced by JSON pointers

2.9.1.10.5.1 Pattern: Provisioning Baseline Package (Conceptual)

```
# Tanium package (conceptual)
# - run baseline collectors over SSH
# - store stdout JSON per-host
run_ssh_identity run_ssh_inventory store_json_outputs
```

2.9.1.10.5.2 Pattern: Monitoring Snapshot Package (Conceptual)

```
# Tanium package (conceptual)
# - run health/reset reason
# - store minimal JSON
run_ssh_health run_ssh_reset_reason store_json_outputs
```

2.9.1.10.5.3 Notes for Puppet and Chef (Drift and Scheduled Execution)

Puppet and Chef are well-suited for:

- Periodic execution of collectors (inventory or monitoring)
- Enforcing baseline presence of tooling
- Scheduled drift detection against known-good baselines

They are not typically used for interactive incident response bundles but can trigger them if desired.

Use Case	Usage
Tooling presence	Ensure the tool directory and dependencies exist on endpoints
Scheduled collectors	Run bounded collectors on a cadence. Forward JSON off-box.
Drift detection	Compare baseline summaries to current summaries and flag divergence
Controllers	Use sparingly. Prefer change-window orchestration platforms for risky actions.
Incident bundles	Possible, but generally better triggered by incident orchestration flows

2.9.2. Custom Installation with cloud-init

By accessing the instructions for custom installation and scaled deployment, the Customer acknowledges that this workflow bypasses individual end-user license prompts. Use of these tools is governed by the *NVIDIA Software License Agreement* (which includes the NVIDIA CUDA EULA). By proceeding, the Administrator represents they have the authority to bind the Customer to these terms and conditions.

2.9.2.1 Overview

This topic describes custom installation workflows for IT administrators and engineers who deploy NVIDIA DGX Spark systems with cloud-init, USB recovery media, and locally hosted Debian packages and firmware. These workflows implement the initial provisioning stage described in *Enterprise Lifecycle Integration*. For cloud-init concepts and where provisioning fits in the fleet lifecycle, refer to *Cloud-init for DGX Spark*.

2.9.2.2 Cloud-init Support

Cloud-init support on DGX Spark Founders Edition is provided by the `nvidia-spark-cloud-init` package, which is included in the October 2026 Release and later. If the system runs an earlier software release, reflash it with an October 2026 Release or later recovery image before using the workflows in this topic. For the recovery download and procedure, refer to *System Recovery*.

For an OEM system, cloud-init is enabled unless the OEM removes it during the image build. Refer to the OEM release documentation for cloud-init availability and recovery-media requirements.

The provided `user-data` examples disable cloud-init after a successful run. If `user-data` creates a user, the initial system setup screens (or out-of-box experience [OOBE]) are skipped. If `user-data` does not create a user, those screens remain available.

2.9.2.3 IT Administrator Workflows

Determine the system software release before you prepare the `OEMDATA` volume:

Device State	Typical First Step	When to Use This Topic
Founders Edition with the October 2026 Release or later	Prepare an `OEMDATA` volume	Use the cloud-init, package, firmware, and local-repository workflows in this topic.
Founders Edition earlier than the October 2026 Release	Create recovery media for the October 2026 Release or later, then add an `OEMDATA` partition	Reflash the system before using cloud-init. Reflashing erases local storage.
OEM system	Follow the OEM release and recovery documentation	Use this topic only when the OEM documentation confirms that these October 2026 Release cloud-init workflows apply.

2.9.2.3.1 Which Installation Path to Use

Choose one setup option and one package or firmware option.

To determine whether your device needs re-imaging, refer to *IT Administrator Workflows*. For details about each pattern, refer to *Air-Gapped and Custom Installation Patterns*.

Setup Option	Start Here
Skip OOBE	Use the `user-data` example that creates a user; refer to *Cloud-init Configurations*.
Keep OOBE and provision without creating a user	Use the `user-data` example that does not create a user. Cloud-init runs provisioning from `bootcmd` while OOBE remains available for the user to create an account.

Package or Firmware Option	Start Here
Deliver packages or firmware from USB	*USB and Image Layout* and *USB Partitioning and the OEM-DATA Layout*
Host a curated APT and firmware tree	*Host a Minimal APT Repository and Firmware Tree* and *On the DGX Spark Client: hook.sh and OEMDATA Files*
Mirror Ubuntu ports and LVFS on the installation network	*Mirror the Full Ubuntu Ports and LVFS Content on a Server*

2.9.2.4 Prerequisites and Source Media

Obtain the resources required by your selected workflow:

Resource	Format	Source
DGX Spark recovery image (required for systems earlier than the October 2026 Release)	`.tar.gz`	DGX Spark System Recovery Image (refer also to *System Recovery*). The download page lists the current recovery-image file name.
Cloud-init seed and hook	Directory tree	`meta-data` identifies the cloud-init instance. `user-data` is the cloud-init configuration and calls optional `hook.sh`. `hook.sh` installs Debian packages and firmware. The examples are in *Reference: Scripts and cloud-init*.
Debian packages (optional)	`.deb`	Debian packages that you supply from public repositories, an OEM, or packages you build or approve. This topic does not provide a package download portal. Place `.deb` files on `OEMDATA` or in a minimal repository you host per *Host a Minimal APT Repository and Firmware Tree*.
Firmware (optional)	`.cab`/`.cap` or LVFS-style mirror content	Firmware that you supply from public repositories, an OEM or ODM, or packages you build or approve. This topic does not provide a firmware download portal. Place firmware on `OEMDATA` or in a minimal tree you host per *Host a Minimal APT Repository and Firmware Tree*. For current Founders Edition component versions, refer to *DGX Spark Release Notes*.

Note

These workflows apply to Founders Edition systems running the October 2026 Release or later. For an OEM system, follow the OEM release documentation.

2.9.2.4.1 Host Tools

Install the tools required by the procedure you follow:

- USB `OEMDATA` partition: `gdisk` (for `sgdisk`), `dosfstools` (for `mkfs.vfat`), and `util-linux` (for `lsblk` and `wipefs`).
- Minimal APT repository: `dpkg-dev`, which provides `dpkg-scanpackages`.
- HTTP server for a minimal repository or `~/mirror`: `python3`.
- Mirror script download and URL checks: `wget` or `curl`.
- Optional `guids.txt` workflow, and the final firmware check in `hook.sh`: `jq`.
- Firmware collection and installation on DGX Spark: `fwupdtool` and `fwupdmgr`.
- Full Ubuntu ports and LVFS mirror: `apt-mirror` and the Python packages listed in that procedure.

2.9.2.5 USB and Image Layout

This section shows the `OEMDATA` USB layout. Use it for the cloud-init seed in every workflow, and when you deliver packages or firmware from USB. Refer to *USB Partitioning and the OEMDATA Layout* for the procedure.

For a system that already runs the October 2026 Release or later, a USB drive can contain one FAT32 partition labeled `OEMDATA`. At minimum, place `meta-data` and `user-data` in the partition root. The following example also includes the optional `hook.sh` file:

```
$ lsblk -o NAME,SIZE,FSTYPE,LABEL,PARTLABEL /dev/sdb
NAME    SIZE FSTYPE LABEL   PARTLABEL
sdb    57.8G
  sdb1 57.7G vfat   OEMDATA OEMDATA

$ sudo mount /dev/sdb1 /mnt/oemdata
$ tree /mnt/oemdata
/mnt/oemdata
    hook.sh
    meta-data
    user-data
```

For a system earlier than the October 2026 Release, first create recovery media for the October 2026 Release or later. Then add a FAT32 `OEMDATA` partition. The resulting USB drive has a recovery partition and an `OEMDATA` partition:

```
$ lsblk -o NAME,SIZE,FSTYPE,LABEL,PARTLABEL /dev/sdb
NAME    SIZE FSTYPE LABEL                 PARTLABEL
sdb    57.8G
  sdb1   31G vfat   BOOTME
  sdb2 26.7G vfat   OEMDATA              OEMDATA
```

You can also host packages and firmware on a network server.

Host curated APT packages and firmware on a host reachable from the installation network. Use the optional URL files on `OEMDATA` to point the client at that server.

2.9.2.6 First Boot Behavior

All of the following conditions must be met for DGX Spark to use the setup volume. If a condition is not met, the system boots normally and ignores the volume without reporting or logging an error:

- **Attach the volume before power-on.** The systemd generator decides whether to run cloud-init early in the boot process. A volume attached during boot, at the login screen, or after login is not detected.
- **Format the partition as FAT32 or ISO 9660.** Cloud-init accepts the label match only on a `vfat` or `iso9660` filesystem.
- **Use the uppercase label** `OEMDATA`. The label is case sensitive.
- **Place both seed files in the partition root.** Both `meta-data` and `user-data` are required, even when `meta-data` only sets an instance ID.

If `user-data` creates a user, cloud-init completes automated setup and OOBE is skipped. If `user-data` does not create a user, provisioning runs from `bootcmd` while OOBE remains available for the user to create an account. When OOBE runs, `runcmd` does not run.

When present, `hook.sh` installs packages and firmware from `debs/` and `firmware/` and configures APT and LVFS sources from the optional URL files.

2.9.2.7 Where to Place Customization Content

The following table summarizes where each type of content belongs. Place `meta-data` and `user-data` on the USB `OEMDATA` volume for every workflow. Do not host those two files on the network server.

Content	USB OEMDATA Partition	Network Server
`meta-data` and `user-data`	Required — partition root	No — USB only
`hook.sh`	Optional — partition root	Optional — keep-OOBE network fallback
`.deb` packages	Optional — `debs/`	Yes — served APT tree
Firmware (`.cab`/`.cap`)	Optional — `firmware/`	Yes — LVFS mirror or signbinpack directory
`apt-repo.url`	Optional	N/A (points to server)
`apt-packages.txt`	Optional	N/A
`lvfs-mirror.url`	Optional	N/A (points to server)

2.9.2.8 OEMDATA URL and List File Reference

When you use `OEMDATA` with a network mirror or curated local server, `hook.sh` reads the following optional files from the `OEMDATA` partition root.

File	Required	Format and Behavior
`apt-repo.url`	Optional	**Format:** One line containing the base URL of the APT repository. The URL must match the path your web server serves (the parent of `Packages.gz`). With a unified server layout, the web root typically contains both `deb-repo/` and `lvfs-mirror/` subdirectories; place APT packages under `deb-repo/` (you can rename that folder). Use the full path—for example, `http://192.168.1.1:8080/deb-repo/`. Do not use the bare host root as `apt-repo.url`. `hook.sh` records this value in `/etc/apt/sources.list.d/oem-local.list`. **If omitted:** `hook.sh` does not configure this additional OEM APT source. The script still configures the example full mirror specified by `MIRROR_SERVER_IP` and `MIRROR_SERVER_PORT`.
`apt-packages.txt`	Optional	**Format:** One package name or `.deb` file name per line. **Example:** `nvidia-spark-ota-check`. **If omitted but ``apt-repo.url`` is present:** `hook.sh` adds the OEM source and runs a single-source `apt upgrade` against that repository only.
`lvfs-mirror.url`	Optional	**Format:** One line containing the base URL of the firmware directory. Point at a directory of `.cab`/`.cap` files that you supply—for example, `http://192.168.1.1:8080/lvfs-mirror/signbinpack-2.152.3-release/`. `hook.sh` reads the directory listing and runs `fwupdmgr install` for each file. **If omitted:** `hook.sh` skips this additional firmware directory. The script still configures the example full mirror specified by `MIRROR_SERVER_IP` and `MIRROR_SERVER_PORT`.

The following examples match these formats:

```
http://192.168.1.1:8080/lvfs-mirror/signbinpack-2.152.3-release
```

```
http://192.168.1.1:8080/deb-repo/
```

```
nvidia-spark-ota-check
```

2.9.2.8.1 Example Hardware Connection

The following figure shows an example hardware topology and workflow for the cloud-init customization process. It illustrates USB preparation, the USB `OEMDATA` layout, the DGX Spark device under test (DUT), and a local server that hosts curated or mirrored APT packages and firmware.

The figure uses the following color legend:

- Yellow: Cloud-init files and USB preparation
- Pink: USB `OEMDATA` partition layout and files
- Blue: DGX Spark DUT

- Aqua: Local server hosting curated or mirrored APT and firmware content
- Switch: Network switch that connects the DGX Spark systems
- Red: Public upstream servers (APT and LVFS)

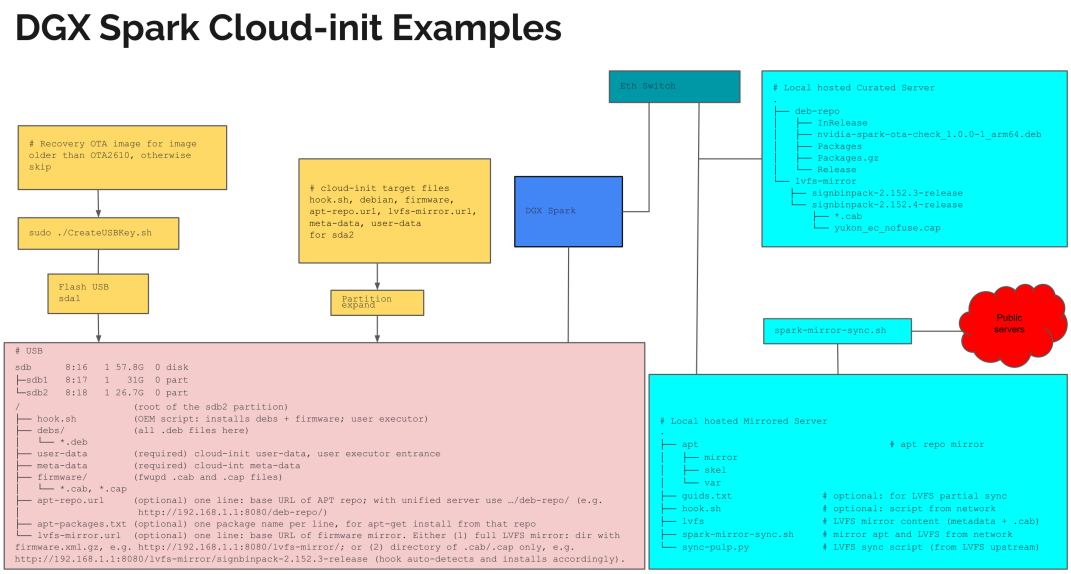


Fig. 3: Spark custom installation: USB `OEMDATA`, optional local server, and client integration.

2.9.2.9 Air-Gapped and Custom Installation Patterns

The following patterns describe how cloud-init, USB layout, and optional local services combine. They align with enterprise customization workflows that use cloud-init OEM seeds and, when needed, an `OEMDATA` partition on the USB device.

Installation and Update Patterns

The following table summarizes the installation and update patterns described in this topic:

Pattern	What You Configure	OEMDATA Partition	Related Sections
Skip OOBE	`user-data` on `OEMDATA` creates the user and runs the provisioning workflow. OOBE is skipped.	Required	*Cloud-init Configurations* and *USB Partitioning and the OEMDATA Layout*.
Keep OOBE and provision without creating a user	`user-data` on `OEMDATA` runs provisioning from `bootcmd` without creating a user. The user creates an account through OOBE.	Required	Same cloud-init and USB references as the row above.
USB-hosted packages and firmware	The `OEMDATA` partition contains the required seed files and optional `hook.sh`, `debs/`, and `firmware/` content. Cloud-init runs `hook.sh` on first boot while the USB drive is present.	Required	*USB Partitioning and the OEMDATA Layout*.
Local server with mirrored (MIRRORED) public sources	A separate host mirrors upstream Ubuntu ports and LVFS content (for example, using `spark-mirror-sync.sh` and related steps), then serves them over HTTP. `OEMDATA` includes `hook.sh` so the client is configured to use that mirror (the sync script runs on the mirror server, not on the USB drive).	Required (for `hook.sh`-based client wiring from USB)	*Mirror the Full Ubuntu Ports and LVFS Content on a Server*.

How the Pieces Fit

The following sequence illustrates a typical deployment workflow:

1. If the DGX Spark runs a release earlier than the October 2026 Release, reflash it with an October 2026 Release or later recovery image.
2. Prepare an `OEMDATA` volume, with or without an APT or firmware server.
3. Power off the DGX Spark, attach the `OEMDATA` volume, and power on the system.
4. Apply configuration with cloud-init. Configuration can include network settings, user accounts, firmware updates, APT updates, and related tasks.
    1. Store configuration and firmware or APT content on the `OEMDATA` volume.
    2. Store configuration on the `OEMDATA` volume and host curated firmware and APT sources on a server.
    3. Host all firmware and APT configuration on the server.

Select and adapt one of these options for your deployment. The examples in this topic provide a framework that you can extend to match your site requirements.

For USB partition layout, content placement, and first-boot behavior, refer to *USB and Image Layout*, *Where to Place Customization Content*, and *First Boot Behavior*. `hook.sh` is reference material for the USB and server layouts described in this topic. If your layout changes, adapt `hook.sh` accordingly.

2.9.2.10 Example Constants

The following tables list example IP addresses, ports, and directory paths. Replace these values with settings that match your environment. In particular, `192.168.1.1:8080` is the example address for the local server. Replace the address in `user-data` and `hook.sh` with the IP address and port for your local server.

> **Warning**
>
> Do not use the default username and password in production deployments. Remove the default credentials or replace them with credentials approved by your organization.

Example Constants for the Repository Server (Port 8080 or 80)

Item	Example Value
Desktop or server IP	`192.168.1.1`
Python HTTP server port	`8080`
Web root	`/var/www` or for example `$HOME/oem-server`
APT subdirectory	`deb-repo` under `WEB_ROOT`
LVFS subdirectory	`lvfs-mirror` under `WEB_ROOT`

2.9.2.11 USB Partitioning and the OEMDATA Layout

Use a FAT32 partition labeled `OEMDATA` for the cloud-init seed and optional packages, firmware, and repository URL files. Both `meta-data` and `user-data` are required in the partition root.

2.9.2.11.1 Create an OEMDATA-Only USB Drive

For a system that already runs the October 2026 Release or later, create one FAT32 partition on a USB drive:

> **Warning**
>
> The following procedure removes existing filesystem signatures from the selected partition. Verify the device and partition names with `lsblk` before running `wipefs` or `mkfs.vfat`. Selecting the wrong device can erase data.

1. Create a partition that uses the available space on the USB drive.
2. Format the partition as FAT32 and label it `OEMDATA`:

```
$ sudo wipefs -a /dev/<partition>
$ sudo mkfs.vfat -F 32 -n OEMDATA /dev/<partition>
```

3. Mount the partition and copy the required seed files:

```
$ sudo mkdir -p /mnt/oemdata
$ sudo mount /dev/<partition> /mnt/oemdata
```

(continues on next page)

(continued from previous page)

```
$ sudo install -m 0644 <seed-directory>/meta-data /mnt/oemdata/meta-data
$ sudo install -m 0644 <seed-directory>/user-data /mnt/oemdata/user-data
```

4. Copy optional `hook.sh`, package, firmware, and URL files as required by your workflow.
5. Run `sync`, unmount the partition, and confirm the layout:

```
$ sync
$ sudo umount /mnt/oemdata
$ lsblk -o NAME,SIZE,FSTYPE,LABEL,PARTLABEL /dev/<USB-device>
```

2.9.2.11.2 Add OEMDATA to Recovery Media

If the system runs a release earlier than the October 2026 Release, create an October 2026 Release or later recovery USB drive first. Refer to *System Recovery* for the recovery-image download and `CreateUSBKey.sh` procedure. Then add `OEMDATA` as partition 2.

The following reference script creates and populates partition 2. Edit the variables before you run it. `DISK` must identify the whole USB device, not a partition.

```
#!/bin/bash
# Add an OEMDATA partition to a USB drive that contains a recovery image.
# Edit the variables, then run with sudo.
set -eu

SEED=<path-to-oemdata>  # Directory containing meta-data and user-data.
DISK=/dev/<USB-device>  # Whole USB drive, not a partition.
DEBS_PATH=              # Optional directory of .deb files.
FW_PATH=                # Optional directory of .cab/.cap files.
URL_FILES=              # Optional URL and package-list files.
MNT=/mnt/oemdata

[ "$(id -u)" -eq 0 ] || { echo "Run with sudo." >&2; exit 1; }
read -r rm tran < <(lsblk -dno RM,TRAN "$DISK")
[ "$rm" = 1 ] || [ "$tran" = usb ] || {
  echo "$DISK is not a removable USB drive." >&2
  exit 1
}
case "$DISK" in
  *[0-9]) PART="${DISK}p2" ;;
  *) PART="${DISK}2" ;;
esac

lsblk -o NAME,SIZE,FSTYPE,LABEL,PARTLABEL,TRAN "$DISK"
read -rp "Create $PART and format it FAT32 as OEMDATA? Type $DISK to confirm:
↪" ok
[ "$ok" = "$DISK" ] || { echo "Aborted."; exit 1; }

if [ ! -b "$PART" ]; then
  # 2:0:0 uses the first aligned sector of the largest free block
  # through the end of that block.
  sgdisk -n 2:0:0 -t 2:0700 -c 2:OEMDATA "$DISK"
  partprobe "$DISK" || true
```

(continues on next page)

(continued from previous page)

```
  udevadm settle || true
fi
[ -b "$PART" ] || { echo "$PART did not appear." >&2; exit 1; }

umount "$PART" 2>/dev/null || true
wipefs -a "$PART"
mkfs.vfat -F 32 -n OEMDATA "$PART"
mkdir -p "$MNT"
mount "$PART" "$MNT"

install -m 0644 "$SEED/meta-data" "$MNT/meta-data"
install -m 0644 "$SEED/user-data" "$MNT/user-data"
[ ! -f "$SEED/hook.sh" ] || install -m 0755 "$SEED/hook.sh" "$MNT/hook.sh"
for f in $URL_FILES; do install -m 0644 "$f" "$MNT/"; done
[ -z "$DEBS_PATH" ] || {
  mkdir -p "$MNT/debs"
  cp "$DEBS_PATH"/*.deb "$MNT/debs/"
}
[ -z "$FW_PATH" ] || {
  mkdir -p "$MNT/firmware"
  cp "$FW_PATH"/*.ca[bp] "$MNT/firmware/"
}

sync
ls -la "$MNT"
umount "$MNT"
lsblk -o NAME,SIZE,FSTYPE,LABEL,PARTLABEL "$DISK"
echo "OEMDATA ready. Power off DGX Spark, attach the drive, then power on."
```

2.9.2.11.3 OEMDATA Partition Layout

The following tree shows all supported files. Only `meta-data` and `user-data` are mandatory:

```
.
    apt-packages.txt    # Optional package list.
    apt-repo.url        # Optional APT repository URL for hook.sh.
    debs/               # Optional .deb files.
    firmware/           # Optional .cab/.cap files.
    hook.sh             # Optional extended script.
    lvfs-mirror.url     # Optional firmware repository URL for hook.sh.
    meta-data           # Required cloud-init meta-data.
    user-data           # Required cloud-init user-data.
```

For URL-file formats, refer to *OEMDATA URL and List File Reference*.

2.9.2.12 Host a Minimal APT Repository and Firmware Tree

This section describes how to host a minimal APT repository and firmware tree on a desktop or server. Use this workflow when you have a curated set of packages and firmware to serve and you do not need a full Ubuntu ports mirror or an LVFS archive mirror. In this context, *minimal* means you index your Debian packages with `dpkg-scanpackages`, host those packages and your firmware in a compact directory layout, and point clients at that layout over HTTP. The steps here assume packages

and firmware are trusted, as in many air-gapped installations, and they do not cover hardening for an internet-exposed package mirror. Refer to *Security Considerations* for risks and mitigations.

One network resource can serve both the APT repository and the LVFS-related tree. Use a single web root (`WEB_ROOT`) with `REPO_DIR` and `LVFS_DIR` as subdirectories.

Define the web root and repository directories with the following commands:

```
WEB_ROOT=/var/www   # or for example $HOME/oem-server
REPO_DIR="$WEB_ROOT/deb-repo"
LVFS_DIR="$WEB_ROOT/lvfs-mirror"
sudo mkdir -p "$REPO_DIR" "$LVFS_DIR"
sudo chown "$USER" "$REPO_DIR" "$LVFS_DIR"
```

2.9.2.12.1 On a Desktop or Server

Complete the following steps to host a minimal APT repository and firmware tree on a desktop or server system:

1. Install tools for the APT repository: `sudo apt-get install -y dpkg-dev`
2. Copy `.deb` files into `REPO_DIR`.
3. Generate the APT index (`Packages.gz`). Re-run this step whenever you add or change `.deb` files. Run the following commands:

```
cd "$REPO_DIR"
dpkg-scanpackages . /dev/null | gzip -9c > Packages.gz
```

4. Optional: Add `Release` and uncompressed `Packages` to avoid 404 responses. `apt` might request `Release`, `Packages` (uncompressed), and similar. A repository that only has `Packages.gz` can return 404 responses that `apt` can tolerate when `Packages.gz` is present. To serve these files, run the following commands:

```
cd "$REPO_DIR"
zcat Packages.gz > Packages 2>/dev/null || gzip -dc Packages.gz > Packages
# Minimal Release file (paths relative to repo root); example block:
{
  echo "Origin: OEM Local Repo"
  echo "Label: oem-local"
  echo "Suite: ."
  echo "Codename: ."
  echo "Architectures: arm64 amd64"
  echo "Components: ."
  echo "Description: OEM local package repository"
  echo "Date: $(date -u -R)"
  echo "MD5Sum:"
  printf ' %s %s Packages.gz\n' "$(md5sum Packages.gz | awk '{print $1}')" "
↪$(stat -c%s Packages.gz)"
  printf ' %s %s Packages\n' "$(md5sum Packages | awk '{print $1}')" "$(stat -
↪c%s Packages)"
  echo "SHA256:"
  printf ' %s %s Packages.gz\n' "$(sha256sum Packages.gz | awk '{print $1}')"
↪"$(stat -c%s Packages.gz)"
  printf ' %s %s Packages\n' "$(sha256sum Packages | awk '{print $1}')" "
↪$(stat -c%s Packages)"
```

(continues on next page)

(continued from previous page)

```
} > Release
cp Release InRelease
```

Whenever you regenerate `Packages.gz` in step 3, repeat step 4: recreate uncompressed `Packages`, write `Release`, and copy `InRelease` using the commands in the code block above.

5. Serve both APT and LVFS over HTTP from one server.

Option A (Python): Run the following command from `WEB_ROOT` to start a simple HTTP server:

```
cd "$WEB_ROOT"
python3 -m http.server 8080 --bind 0.0.0.0
```

Option B (Nginx): Install and enable Nginx, then add a configuration similar to the following example:

```
location /deb-repo {
  alias /var/www/deb-repo;
  autoindex on;
}
location /lvfs-mirror {
  alias /var/www/lvfs-mirror;
  autoindex on;
}
```

With this layout, the server root lists only `deb-repo/` and `lvfs-mirror/`. On the USB drive you must use full paths, not the server root alone.

The following examples show typical APT and LVFS URLs:

- APT repository URL: `http://192.168.1.1:8080/deb-repo/` (Python) or `http://192.168.1.1/deb-repo/` (Nginx).
- Firmware directory URL: a subdirectory of `.cab`/`.cap` files, for example `http://192.168.1.1:8080/lvfs-mirror/signbinpack-2.152.3-release/` (`hook.sh` auto-detects and installs each file).

6. Firewall: allow inbound HTTP on the port you use (for example, `sudo ufw allow 80/tcp`, `sudo ufw allow 8080/tcp`, `sudo ufw reload`).

2.9.2.12.2 On the DGX Spark Client: hook.sh and OEMDATA Files

Place the following files on the USB drive `OEMDATA` partition when you want `hook.sh` on the client to use your hosted APT repository. For file format rules, refer to *OEMDATA URL and List File Reference*.

- `apt-repo.url`: Include this file when the client should use your hosted APT repository.
- `apt-packages.txt`: Optional. If present, one package per line. Each line can be either a package name (for example, `nvidia-spark-ota-check`) or a full `.deb` file name (for example, `nvidia-spark-ota-check_1.0.0-1_arm64.deb`); the hook derives the package name from a `.deb` file name when needed and runs `apt-get install` for that set.

With `apt-repo.url` present, `hook.sh` wires `oem-local.list`, updates the index, then either installs listed packages from `apt-packages.txt` or performs the single-source upgrade when `apt-packages.txt` is absent. Refer to the listing in *First Boot: OEMDATA hook.sh and cloud-init Seed* or `enterprise-custom-install-oemdata-hook.sh` in this documentation source.

Troubleshooting: Use the following notes when you diagnose APT repository issues on the client:

- Ignore or 404 for `Release.gpg` and `InRelease`: Expected for an unsigned repository; `[trusted=yes]` makes `apt` ignore the missing signature. Refer to *Security Considerations* for the risks of unsigned local repositories.
- "Unable to locate package": The repository is added (`/etc/apt/sources.list.d/oem-local.list`). `apt` fetches `Release` but might not load `Packages` if the `Release` file is wrong. Ensure that `Release` has `Date`, paths `Packages.gz` and `Packages`, and run `cp Release InRelease` (step 4 above). The hook's fallback (download `.deb` and `dpkg -i`) works even when `apt` does not see the package.

2.9.2.12.3 Minimal LVFS Mirror on the Same Host

Use `LVFS_DIR` under the same `WEB_ROOT` as the APT repository. `hook.sh` can install firmware from this tree over the LAN when `lvfs-mirror.url` points at a directory of `.cab/.cap` files, in addition to any `.cab/.cap` from the USB `firmware/` directory.

2.9.2.12.3.1 On a Desktop (Server)

Complete the following steps to populate and serve a minimal LVFS mirror on the same host as the APT repository:

1. Populate `LVFS_DIR` with firmware content that you supply (for example, from a public repository, an OEM or ODM, or packages you build or approve). Place firmware files (for example, from a signbinpack release) in a subdirectory such as `$LVFS_DIR/signbinpack-2.152.3-release/`. Point `lvfs-mirror.url` on the client at that subdirectory's URL. `hook.sh` reads the directory listing and runs `fwupdmgr install` for each `.cab/.cap` file.
2. Serve the mirror over HTTP using the same web server as the APT repository (run from `WEB_ROOT`). For example: `http://192.168.1.1:8080/lvfs-mirror/signbinpack-2.152.3-release/` (Python) or `http://192.168.1.1/lvfs-mirror/signbinpack-2.152.3-release/` (Nginx).
3. On the host that serves both trees, allow inbound HTTP on the ports you use for that server (typically the same ports you opened for the APT repository). Run the following commands, for example:

```
sudo ufw allow 80/tcp
sudo ufw allow 8080/tcp
sudo ufw reload
```

2.9.2.12.3.2 On the DGX Spark Client

Place the following file on the USB drive `OEMDATA` partition when you want `hook.sh` on the client to use your hosted firmware mirror:

- `lvfs-mirror.url`: Include a single line with the base URL of the firmware directory. When the URL points to a directory of `.cab/.cap` files (for example, a signbinpack release subdirectory), the hook fetches the directory listing, downloads each file, and runs `fwupdmgr install` for each one.

By default, `fwupd` installs only trusted (LVFS-signed) firmware. If installation of local or vendor `.cab/.cap` files fails with a message such as "firmware signature missing or not trusted" (for example, signbinpack content from the mirror or from USB `firmware/`), edit `/etc/fwupd/fwupd.conf` on the client and set `OnlyTrusted=false` under `[fwupd]`: The following example shows the required setting:

```
[fwupd]
OnlyTrusted=false
```

Note

Use `OnlyTrusted=false` only when you control the firmware source and accept the risk. Refer to *Security Considerations* for related firmware-mirror guidance.

2.9.2.13 Mirror the Full Ubuntu Ports and LVFS Content on a Server

Use this workflow when you need Ubuntu ports and LVFS content on the installation network and can host a large mirror tree. The following procedures create a unified layout for Ubuntu ports APT repositories and LVFS content under `~/mirror`. In the examples, Spark A acts as the mirror server and serves the content over HTTP on port 8080; Spark B acts as the client.

The mirror server can be another DGX Spark or an x86 system. The example scripts were verified on DGX Spark. An x86 server might require changes. The APT mirror example adds only the `noble-proposed` repository by default. To mirror additional repositories, update the script by following the same pattern used for `noble-proposed`.

2.9.2.13.1 Server Directory Layout

The following example shows the expected directory tree under the mirror root (for example, `tree -L 2 ~/mirror`):

```
.
    apt
        mirror
        skel
        var
    guids.txt              # optional: LVFS partial sync (--guid-file)
    hook.sh
    lvfs                   # LVFS mirror (metadata + .cab)
    spark-mirror-sync.sh   # One-shot mirror script
    sync-pulp.py           # LVFS sync script (from LVFS upstream)
```

Client URL Patterns (Must Match Layout)

The following table shows the client URL patterns that must match the server layout:

Service	URL Pattern
apt	`http://SERVER_IP:8080/apt/mirror/ports.ubuntu.com/ubuntu-ports/`
fwupd	`MetadataURI=http://SERVER_IP:8080/lvfs/<metadata-file>` and `FirmwareBaseURI=http://SERVER_IP:8080/lvfs`

If you rename `lvfs`, change both `MetadataURI` and `FirmwareBaseURI` on clients to match.

2.9.2.13.2 Create the Top-Level Tree and sync-pulp.py

Create the mirror directory tree and download `sync-pulp.py` with the following commands:

```
mkdir -p ~/mirror/apt ~/mirror/lvfs
cd ~/mirror
wget -O sync-pulp.py https://gitlab.com/fwupd/lvfs-website/raw/master/contrib/
↪sync-pulp.py
chmod +x sync-pulp.py
```

Create `guids.txt` only if you plan a partial LVFS sync. For the sync command, refer to *Partial Mirror (GUID File, ~50 MB)*. For file generation instructions and formatting rules, refer to *guids.txt for a Partial LVFS Sync*.

2.9.2.13.3 One-Shot Sync Script: spark-mirror-sync.sh

Copy `oemdata/spark-mirror-sync.sh` from your distribution package onto the Spark, or run it from a repository clone. Some trees place this file under `scripts/`; use the path that matches your bundle. Run as root (`sudo`); the script does not invoke `sudo` internally.

The script mirrors both APT and LVFS content. It was verified on DGX Spark and might require changes on an x86 server. The APT examples in this guide use `noble-proposed` only. Add other suites or repositories to the script if your deployment requires them.

The script performs the following actions:

- Installs dependencies only if you pass `--install-deps` / `--install-apt-mirror`.
- Creates `${MIRROR_ROOT}/apt-mirror.list.spark` if missing (noble-proposed, `base_path` = `$MIRROR_ROOT/apt`).
- Runs `apt-mirror`, then `sync-pulp.py` into `$MIRROR_ROOT/lvfs`.
- Symlinks `/usr/local/bin/python` to `python3` for tools that expect `python`, and runs `sync-pulp.py` with `python3`.

Run `spark-mirror-sync.sh` with environment variables set as shown in the following examples:

```
export LVFS_USERNAME='you@example.com'
export LVFS_TOKEN='your-lvfs-token'
sudo -E ./spark-mirror-sync.sh --install-deps --install-apt-mirror   # first
↪run only
sudo -E ./spark-mirror-sync.sh --install-deps --install-apt-mirror --skip-
↪lvfs   # skip LVFS; sync separately
sudo -E ./spark-mirror-sync.sh
```

Default `MIRROR_ROOT` with `sudo` and without `-H` is `/root/mirror`. To mirror under a user home directory (for example, `/home/nvidia/mirror`), run the following command:

```
sudo env MIRROR_ROOT=/home/nvidia/mirror ./spark-mirror-sync.sh
```

Optional: `APT_MIRROR_LIST`, `--skip-apt`, `--skip-lvfs`, `LVFS_CLEANUP=1` for `--cleanup` on LVFS. If `$MIRROR_ROOT/guids.txt` exists, the script passes `--guid-file` automatically.

2.9.2.13.4 APT Mirror (noble-proposed under ~/mirror/apt)

If a manual APT mirror is required, this section provides an example workflow for creating one. The packaged `/usr/bin/apt-mirror` on Ubuntu is often too old to mirror some DEP-11 paths (for example, `icons-64x64@2.tar`). Install the current upstream `apt-mirror` Perl script from GitHub with the following commands:

```
sudo apt install -y perl wget
sudo cp -a /usr/bin/apt-mirror /usr/bin/apt-mirror.distpkg 2>/dev/null || true
sudo wget -O /usr/local/bin/apt-mirror https://raw.githubusercontent.com/apt-
↪mirror/apt-mirror/master/apt-mirror
sudo chmod +x /usr/local/bin/apt-mirror
```

Always run synchronization with `/usr/local/bin/apt-mirror`.

`apt-mirror` stores the Ubuntu tree under `$base_path/mirror/`. With `base_path` set to `~/mirror/apt`, the live archive path is `~/mirror/apt/mirror/ports.ubuntu.com/ubuntu-ports/`, matching the client URI after `http://SERVER_IP:8080/apt/`.

Create the mirror list file that `/usr/local/bin/apt-mirror` uses for the mirror configuration. The following example uses `/home/nvidia/mirror/apt` as the base path:

```
set base_path /home/nvidia/mirror/apt

# Only noble-proposed for ports.ubuntu.com/ubuntu-ports (+ optional deb-src
↪if you mirror sources)

clean http://ports.ubuntu.com/ubuntu-ports
sudo /usr/local/bin/apt-mirror /etc/apt/mirror.spark.list
```

Re-run periodically (for example, through `cron`) when you need fresher packages. Allow **8080/tcp** from client subnets if a host firewall is enabled.

2.9.2.13.5 The LVFS (fwupd) Mirror Under ~/mirror/lvfs

Install the Python dependencies for `sync-pulp.py` with the following command:

```
sudo apt install -y python3 python3-requests python3-lxml
```

Full Mirror (Large, on the Order of ~300 GB)

A full mirror requires an LVFS account and user token (not your account password). Refer to the LVFS site for account and token issuance. Run the following command to synchronize the full LVFS tree:

```
cd ~/mirror
./sync-pulp.py https://fwupd.org/downloads ~/mirror/lvfs \
  --username='your-email@example.com' \
  --token='YOUR_USER_TOKEN'
```

Re-run to update; existing valid files are skipped.

Optional: `--cleanup` removes files no longer in the manifest.

Partial Mirror (GUID File, ~50 MB)

On a representative Spark, run `sudo fwupdtool get-devices` or `fwupdmgr get-devices --show-all`. Build `~/mirror/guids.txt`, then run the following command:

```
./sync-pulp.py https://fwupd.org/downloads ~/mirror/lvfs \
  --username='your-email@example.com' \
  --token='YOUR_USER_TOKEN' \
  --guid-file=guids.txt
```

2.9.2.13.6 Patch After Mirror: Choose MetadataURI for Clients

After synchronization, choose a metadata file that has a matching `.jcat` file with the same base-name (for example, `firmware.xml.xz` and `firmware.xml.xz.jcat`, or a versioned pair such as `firmware-08681-stable.xml.xz` and `firmware-08681-stable.xml.xz.jcat`). Use the newest stable pair that exists on disk. Partial mirrors from `--guid-file` often omit top-level `firmware.xml.*` metadata. Run the following safety-net script on the server before configuring clients:

```
LVFS_DIR="$HOME/mirror/lvfs"
LVFS_CDN="https://cdn.fwupd.org/downloads"
for ext in xz gz zst; do
  meta="${LVFS_DIR}/firmware.xml.${ext}"
  jcat="${meta}.jcat"
  if [[ ! -s "${meta}" ]]; then
    echo "Safety-net: ${meta##*/} missing; fetching from ${LVFS_CDN}"
    if curl -fSL --connect-timeout 10 --max-time 120 \
      -o "${meta}.new" "${LVFS_CDN}/firmware.xml.${ext}" \
      && curl -fSL --connect-timeout 10 --max-time 30 \
      -o "${jcat}.new" "${LVFS_CDN}/firmware.xml.${ext}.jcat"; then
      mv -f "${meta}.new" "${meta}"
      mv -f "${jcat}.new" "${jcat}"
    else
      rm -f "${meta}.new" "${jcat}.new"
      echo "WARN: failed to fetch firmware.xml.${ext} / .jcat from ${LVFS_CDN}
↪" >&2
    fi
  fi
done
```

Run the following commands on the server to confirm that metadata and signature files are present:

```
cd ~/mirror/lvfs
for f in firmware*.xml.xz firmware*.xml.gz firmware*.xml.zst; do
  [ -f "$f" ] || continue
  [ -f "${f}.jcat" ] && echo "OK: $f"
done

# below three messages should be observed
OK: firmware.xml.xz
OK: firmware.xml.gz
OK: firmware.xml.zst
```

2.9.2.13.7 guids.txt for a Partial LVFS Sync

For a partial LVFS sync, create `guids.txt` with the GUIDs of the devices whose firmware you want to mirror.

2.9.2.13.7.1 Generate guids.txt

To create the file manually, save the `fwupdtool get-devices` output and copy only UUIDs in the format `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` into `~/mirror/guids.txt`. Run `sort -u ~/mirror/guids.txt -o ~/mirror/guids.txt` to sort the file and remove duplicate lines.

Alternatively, use the following JSON workflow from your user session to extract GUIDs from `fwupdmgr` output. This command requires `jq`:

```
fwupdmgr get-devices --json | jq -r '
  .. | objects | select(has("Guid")) | .Guid,
  (.. | objects | select(has("Guids")) | .Guids[]?)
' | sort -u > ~/mirror/guids.txt
```

To remove non-UUID characters from `guids.txt`, run the following commands:

```
awk '{ gsub(/[^0-9A-Fa-f-]/, "", $0); if (length($0)) print $0 }' \
    ~/mirror/guids.txt > ~/mirror/guids.txt.clean
mv ~/mirror/guids.txt.clean ~/mirror/guids.txt
```

2.9.2.13.7.2 Formatting Rules

Follow these rules when you create `guids.txt` for a partial LVFS sync:

Rule	Detail
One UUID per line	Use the UUID exactly as printed (lowercase hex is acceptable).
No comments or blank lines	Do not use # comments, hardware hints after , or blank lines.
Include device GUIDs	Copy only the UUIDs from `Guid:` lines and `GUIDs:` blocks for devices whose firmware you want to mirror.
Unique UUIDs only	The same GUID often appears on more than one device (for example, several identical NICs). List each UUID only once.
What to include	Include UUIDs for internal or updatable components whose firmware you want in the mirror (such as EC, TPM, UEFI capsules, NVMe, or dbx).
What to omit	Omit removable USB devices if you do not require LVFS content for that class of hardware.

2.9.2.13.7.3 Example guids.txt Listing

Example guids.txt for NVIDIA DGX Spark (`spark-cr01`, `sudo fwupdtool get-devices`): The following listing reflects a typical Spark (Kingston USB flash drive, EC, four ConnectX-7 ports with the same four GUIDs repeated, Samsung NVMe, TPM, two UEFI ESRT firmware slots, and UEFI dbx), including that USB device:

```
09321615-5d32-5758-8308-52a4a7be8efc
095ba8dd-3778-52b4-9f32-02a67c210ce5
0eb9bda9-3010-493a-a6a8-b5e80eddf870
10ec82f4-ff64-5362-9e5d-688febf5dbb0
12029307-5bb1-5200-99a5-536f1be9d081
```

(continues on next page)

(continued from previous page)

```
35abf34a-7ed8-51b2-ba1b-edef527d47e6
3d13c989-e6a8-4ead-95ee-921f09868f65
59007998-a3d7-54a3-b30e-eb3b77e2f351
5f106816-21fe-5d90-896a-175038b9256f
67d35028-ca5b-5834-834a-f97380381082
75b1af35-b88a-59d2-a3c7-a38537f8607f
93768061-87bf-5c78-b9ea-5b7a6301012b
b488217b-3895-4fc0-b1bf-ab7005a2d45a
b5e95689-ad65-5e57-8778-897f04396256
cfc0de0b-adb3-5060-ba22-e4010a78368f
dd1a238a-5f8e-46bd-9401-a88da99c5a96
```

Smaller Mirror (same machine; omit DT microDuo 3C): Delete these three Kingston-only lines:

- `09321615-5d32-5758-8308-52a4a7be8efc`
- `5f106816-21fe-5d90-896a-175038b9256f`
- `75b1af35-b88a-59d2-a3c7-a38537f8607f`

`sync-pulp.py` cannot combine `--guid-file` and `--filter-tag` in one run; run twice if you need both, or consult LVFS offline documentation. Filtered syncs can occasionally leave orphaned metadata relative to `.jcat` pairs; verify pairs and fetch missing top-level metadata with the safety-net script above if needed.

2.9.2.13.8 Serve ~/mirror With Python

After the mirror sync completes, serve the `~/mirror` tree over HTTP with Python as follows.

```
cd ~/mirror
python3 -m http.server 8080 --bind 0.0.0.0
```

- `apt` on the wire: `http://SERVER_IP:8080/apt/mirror/...`
- LVFS on the wire: `http://SERVER_IP:8080/lvfs/...`

Keep this process running (`tmux`, a systemd user unit, or equivalent) while clients update.

`apt-repo.url` and `lvfs-mirror.url` must point to the correct addresses to use the mirror APT repository and LVFS content.

2.9.2.13.9 Cloud-init Integration

The `user-data` file integrates `hook.sh` with cloud-init during first boot. Place the applicable `user-data` example and `hook.sh` in the `OEMDATA` partition root. If the hook exit code is 1, the example records mirror-setup success, runs `sync`, and schedules a reboot.

2.9.2.14 Security Considerations

Consider the following security implications when you deploy custom installation media, local mirrors, and `hook.sh` automation:

- **Secure Boot:** Before booting recovery or customized installation media, confirm the Secure Boot requirements for your deployment and platform policy. Use only NVIDIA-provided or otherwise approved boot media. If Secure Boot settings are changed for installation or recovery, restore the required Secure Boot state before returning the system to service.

- hook.sh **and USB contents:** The hook runs with elevated privileges and executes content from the OEMDATA partition. Anyone with physical access can replace hook.sh, Debian packages, or firmware on the USB drive. Treat the USB drive as trusted input: use tamper-aware handling, restrict who can prepare USB devices, or verify integrity (for example, hashes or signatures) if your policy requires it.
- **Local APT repository:** The hook adds the repository with [trusted=yes], so packages from that repository are not signature-verified. Ensure the repository server and network are trusted.
- **Local firmware mirror:** Firmware from the mirror (or a directory of .cab/.cap) is installed by fwupd. If you set OnlyTrusted=false, unverified or vendor-signed firmware is allowed only when you control the firmware source and accept the risk.
- **URLs on the USB drive:** The values in apt-repo.url and lvfs-mirror.url identify servers on your network. Those servers must be trustworthy. If an attacker compromises one of those servers, or performs a man-in-the-middle (MITM) attack on the path between the client and the server, the client could install malicious packages or firmware.
- **Network exposure:** The host that serves the APT repository and firmware tree is reachable from other systems on the same local network used for DGX Spark installation. Harden that host (access control, firewall, operating system updates). When your security policy requires it, place installation and repository access on a dedicated or isolated network segment instead of a general-purpose LAN.
- **Account password:** The user-creation example stores the chpasswd password as cleartext in user-data on the FAT32 OEMDATA partition. Anyone with physical access to the volume can read it. Protect the volume and replace the example credentials with values approved by your organization.

2.9.2.15 Verify the Customization and Installation Outcomes

Use the following subsections to confirm that your custom installation, mirror configuration, and first-boot provisioning completed successfully.

2.9.2.15.1 Prove cloud-init runs

Installation of nvidia-spark-cloud-init is logged in /var/log/oem-iso-cfg.log during OS installation. Review this log to confirm that the package was installed.

Cloud-init writes user-data and executor output to /var/log/cloud-init-output.log. Review this log to confirm that the provisioning scripts ran and to diagnose errors.

2.9.2.15.2 After Mirror- or USB-Driven Updates

After first boot completes, confirm that mirror- or USB-driven updates applied successfully:

- Confirm that the expected Debian or Ubuntu packages are installed from the mirror.
- Confirm expected firmware versions after fwupdmgr update, as applicable.
- Inspect cloud-init logs for hook execution and errors.
- Disable cloud-init for subsequent boots if your operational model requires it, per your site policy.

2.9.2.16 Prepare the Installation Media and Client (Verification Flow)

Use the following checklist when validating an end-to-end flow:

1. When you use the curated local repository workflow, host packages and firmware per *Host a Minimal APT Repository and Firmware Tree* and start the web server from `WEB_ROOT`.
2. Prepare the FAT32 `OEMDATA` volume with the required `meta-data` and `user-data` files.
3. Copy `hook.sh` and optional packages, firmware, and URL files to `OEMDATA` as required by your workflow.
4. If the system runs a release earlier than the October 2026 Release, create an October 2026 Release or later recovery media and reflash the system.

> **Warning**
>
> Flashing the client re-images the DGX Spark system and erases local storage. Do not proceed until backups are complete, the bootable media has been verified, and the target system has been identified.

5. Power off the DGX Spark, attach the `OEMDATA` volume, and power on the system.
6. Verify that the DGX Spark boots successfully, cloud-init completed without errors, the expected users and configuration are present, required packages and firmware versions are installed, and the Secure Boot state matches your deployment policy.

2.9.2.17 Reference: Scripts and cloud-init

> **Note**
>
> The sample code uses `YOUR_USERNAME` and `YOUR_PASSWORD` as placeholder values for the DGX Spark account credentials. Before you use the sample code, replace these placeholders with the username and password for your DGX Spark system.

The following listings provide reference copies of `hook.sh` and cloud-init seed files for the October 2026 Release workflow.

The following tree shows the supported `OEMDATA` layout:

```
.
    apt-packages.txt
    apt-repo.url
    debs/
    firmware/
    hook.sh
    lvfs-mirror.url
    meta-data
    user-data
```

2.9.2.17.1 First Boot: OEMDATA hook.sh and cloud-init Seed

hook.sh lives on the OEMDATA partition. The user-creation user-data example mounts that partition, copies the hook to /tmp, exports OEM_MNT, and runs the copy. The keep-OOBE example runs the provisioning worker from bootcmd.

Important

Keep the OEMDATA partition and any required local mirror reachable during first boot. Cloud-init runs the provisioning workflow during this stage and can take several minutes while it installs packages, applies firmware updates, refreshes APT or fwupd metadata, and runs hook.sh. Do not power off the system or disconnect required media until provisioning completes.

2.9.2.17.1.1 oemdata/hook.sh

The following script is the reference hook.sh that runs from the OEMDATA partition or from a network URL during first boot.

What this script does

The script performs the following actions in order:

1. Applies default mirror URL settings and switches the client to the local APT and LVFS mirror endpoints.
2. Installs Debian packages from the USB debs/ directory when present and from the local APT repository when apt-repo.url is present (with optional apt-packages.txt).
3. Installs firmware from the USB firmware/ directory when present and from the URL in lvfs-mirror.url when that file is present.
4. Refreshes fwupd, checks for remaining firmware and APT upgrades, applies available upgrades, then restores stock APT sources and re-enables the public LVFS remote after a successful final pass.

```
#!/bin/sh
# OEM hook script: run from cloud-init when OEMDATA partition is mounted.
# Copy this file to the root of the OEMDATA partition (next to debs/ and
 ↪firmware/).
# Optional: apt-repo.url (full path to repo, e.g. …/deb-repo/), apt-packages.
 ↪txt;
# lvfs-mirror.url (base URL only, dir containing firmware.xml.gz,
# e.g. …/lvfs-mirror/ – not a subpath).
# Exit: 0 = success, no apt/fwupd updates in final pass;
# 1 = success and at least one applied;
# 255 = failure (-1 in 8-bit).
# After a successful final pass, current sources.list.d is renamed to
# sources.list.d.cldnt,
# stock apt is restored from sources.list.d.org, and public LVFS is re-enabled.
# Override: MIRROR_SERVER_IP, MIRROR_SERVER_PORT, LVFS_METADATA_NAME, LVFS_WEB_
 ↪SUBDIR
# OEM_MNT is the OEMDATA partition root (debs/, firmware/, urls).
# Normally the directory containing this script; cloud-init may copy this file
# to /tmp and set OEM_MNT explicitly.
# This tree does not use functions.sh. If you extend the USB copy,
```

(continues on next page)

(continued from previous page)

```
# source helpers only as
#   . "$OEM_MNT/your-helper.sh"
# never . "$(dirname "$0")/..." or files vanish when $0 is under /tmp.

# By USING THIS SCRIPT, USER AGREES TO THE TERMS OF THE EULA,
# SOL is enabled by default.
# TELEMETRY is not enabled by default. If enabling telemetry, user agrees to
# accept TELEMETRY terms.

OEM_MNT=${OEM_MNT:-$(dirname "$0")}
WIFI_ADAPTER=${WIFI_ADAPTER:-wlP9s9} # Default WiFi adapter name

# ------------------------------------------------------------------------
# Defaults / mirror URLs (used by mirror + OEM stages)
# ------------------------------------------------------------------------
_hook_config_defaults() {
 # Example defaults. Replace these values for your installation network.
 MIRROR_SERVER_IP=${MIRROR_SERVER_IP:-192.168.1.1}
 MIRROR_SERVER_PORT=${MIRROR_SERVER_PORT:-8080}
 LVFS_WEB_SUBDIR=${LVFS_WEB_SUBDIR:-lvfs}
 LVFS_METADATA_NAME=${LVFS_METADATA_NAME:-firmware.xml.xz}
 MIRROR_APT_URI="http://${MIRROR_SERVER_IP}:${MIRROR_SERVER_PORT}/apt/mirror/
↪"\
 "ports.ubuntu.com/ubuntu-ports/"
 MIRROR_FW_BASE="http://${MIRROR_SERVER_IP}:${MIRROR_SERVER_PORT}/${LVFS_WEB_
↪SUBDIR}"
}

# ------------------------------------------------------------------------
# HTTP GET to stdout (wget or curl)
# ------------------------------------------------------------------------
_hook_http_get() {
 wget -qO - "$1" 2>/dev/null || curl -sL "$1" 2>/dev/null
}

# ------------------------------------------------------------------------
# OOBE post-steps (EXIT trap): run when user "YOUR_USERNAME" exists;
# never fail the hook.
# ------------------------------------------------------------------------
_hook_oobe_supplementary_groups() {
 usermod -aG adm,sudo,audio,dip,plugdev,users,lpadmin \
  YOUR_USERNAME 2>/dev/null || true
}

_hook_oobe_spark_autostart_and_keyboard() {
 install -d -m 0755 -o YOUR_USERNAME -g YOUR_USERNAME \
  /home/YOUR_USERNAME/.config/autostart 2>/dev/null || true
 if [ ! -f /home/YOUR_USERNAME/.config/autostart/nvidia-spark-docs.desktop ];
↪then
   ( umask 022
     cat > /home/YOUR_USERNAME/.config/autostart/nvidia-spark-docs.desktop <<
↪'EOF'
```

(continues on next page)

(continued from previous page)

```
[Desktop Entry]
Type=Application
Name=NVIDIA Spark documentation
Exec=xdg-open https://build.nvidia.com/spark
X-GNOME-Autostart-enabled=true
EOF
   ) 2>/dev/null || true
   chown YOUR_USERNAME:YOUR_USERNAME \
     /home/YOUR_USERNAME/.config/autostart/nvidia-spark-docs.desktop \
     2>/dev/null || true
   chmod 0644 /home/YOUR_USERNAME/.config/autostart/nvidia-spark-docs.desktop
↪\
     2>/dev/null || true
 fi
 if [ -f "$OEM_MNT/oem-keyboard-spark.sh" ]; then
   echo "[oem hook] running $OEM_MNT/oem-keyboard-spark.sh"
   sh "$OEM_MNT/oem-keyboard-spark.sh" || true
 fi
}

_hook_oobe_skip_gnome_initial_setup() {
 install -d -m 0755 -o YOUR_USERNAME -g YOUR_USERNAME \
  /home/YOUR_USERNAME/.config 2>/dev/null || true
 touch /home/YOUR_USERNAME/.config/gnome-initial-setup-done 2>/dev/null ||
↪true
 chown YOUR_USERNAME:YOUR_USERNAME \
  /home/YOUR_USERNAME/.config/gnome-initial-setup-done 2>/dev/null || true
}

_hook_oobe_hotspot_teardown_if_ethernet() {
 # Run as a child (not ".") so dgx-oobe sees $0 under /opt/nvidia/dgx-oobe
 # (functions.sh path).
 # Use bash: functions.sh uses bash syntax; /bin/sh (dash) errors with
 # "(" unexpected.
 _hs=/opt/nvidia/dgx-oobe/oobe-hotspot-shutdown.sh
 if [ -f "$_hs" ]; then
   command -v bash >/dev/null 2>&1 && bash "$_hs" || true
 fi
}

_hook_oobe_disable_systemd_units() {
 for u in dgx-oobe dgx-oobe-admin dgx-oobe-hotspot dgx-oobe-hostname \
   dgx-oobe-hotspot-watchdog; do
   systemctl stop "$u" 2>/dev/null || true
   systemctl disable "$u" 2>/dev/null || true
 done
 if [ -f /etc/NetworkManager/dnsmasq-shared.d/dgx-oobe.conf ]; then
   rm -f /etc/NetworkManager/dnsmasq-shared.d/dgx-oobe.conf
 fi
 systemctl restart avahi-daemon 2>/dev/null || true
 # Disable WiFi adapter scan
 if [ -z "${WIFI_ADAPTER}" ]; then
```

(continues on next page)

(continued from previous page)

```
    return 0
 fi
 if ip link show ${WIFI_ADAPTER}_scan >/dev/null 2>&1; then
     /usr/bin/ip link set ${WIFI_ADAPTER}_scan down || true
     /usr/sbin/iw dev ${WIFI_ADAPTER}_scan del || true
 fi
}

_hook_oobe_ubuntu_pro_attach() {
 if [ -n "${UBUNTU_PRO_TOKEN:-}" ] && command -v pro >/dev/null 2>&1; then
   pro attach "$UBUNTU_PRO_TOKEN" --no-prompt 2>/dev/null || true
 fi
}

_hook_oobe_sol_if_consent() {
 echo "[oem hook] Enabling SOL"
 install -d -m 0755 /opt/nvidia/dgx-telemetry 2>/dev/null || true
 touch /opt/nvidia/dgx-telemetry/eula_accepted 2>/dev/null || true
 sync
 systemctl daemon-reload 2>/dev/null || true
 if ! systemctl enable --now nvidia-dgx-sol 2>/dev/null; then
   echo "[oem hook] warning: systemctl enable --now nvidia-dgx-sol failed" \
     "(check status; unit may stay disabled)" >&2 || true
   systemctl start nvidia-dgx-sol 2>/dev/null || true
 fi
}

_hook_oobe_telemetry_if_consent() {
 echo "[oem hook] Enabling telemetry"
 install -d -m 0755 /opt/nvidia/dgx-telemetry 2>/dev/null || true
 touch /opt/nvidia/dgx-telemetry/technical_consent \
   /opt/nvidia/dgx-telemetry/functional_consent 2>/dev/null || true
 sync
 systemctl daemon-reload 2>/dev/null || true
 if ! systemctl enable --now nvidia-dgx-telemetry 2>/dev/null; then
   echo "[oem hook] warning: systemctl enable --now nvidia-dgx-telemetry
↪failed" \
     "(check status; unit may stay disabled)" >&2 || true
   systemctl start nvidia-dgx-telemetry 2>/dev/null || true
 fi
}

_hook_oobe_complete_flag_marker() {
 install -d -m 0755 /opt/nvidia/dgx-oobe 2>/dev/null || true
 touch /opt/nvidia/dgx-oobe/oobe-complete-flag 2>/dev/null || true
}

# When cloud-init created user "nvidia", run one-time OOBE-aligned steps on
↪every
# script exit.
# (EXIT runs after normal completion, exit 1, or exit 255 so these steps still
↪run.)
```

(continues on next page)

(continued from previous page)

```
_hook_oobe_post() {
 set +e
 if [ "$(id -u)" -ne 0 ]; then
   return 0
 fi
 if ! getent passwd YOUR_USERNAME >/dev/null 2>&1; then
   return 0
 fi
 echo "[oem hook] OOBE post-steps for user YOUR_USERNAME (EXIT trap)"
 _hook_restore_stock_apt_and_lvfs || true
 _hook_oobe_supplementary_groups || true
 _hook_oobe_spark_autostart_and_keyboard || true
 _hook_oobe_skip_gnome_initial_setup || true
 _hook_oobe_hotspot_teardown_if_ethernet || true
 _hook_oobe_disable_systemd_units || true
 _hook_oobe_ubuntu_pro_attach || true
 _hook_oobe_sol_if_consent || true
 _hook_oobe_complete_flag_marker || true

 return 0
}

trap '_hook_oobe_post' EXIT

# ------------------------------------------------------------------------------
# Unified Spark mirror: local apt + fwupd LVFS remote
# ------------------------------------------------------------------------------
_hook_mirror_archive_stock_sources() {
 if [ ! -d /etc/apt/sources.list.d.org ]; then
   if [ -d /etc/apt/sources.list.d ]; then
     mv /etc/apt/sources.list.d /etc/apt/sources.list.d.org
   fi
 fi
 rm -rf /etc/apt/sources.list.d
 mkdir -p /etc/apt/sources.list.d
}

_hook_mirror_write_deb822_sources() {
 cat > /etc/apt/sources.list.d/local-mirror.sources <<EOF
# Ubuntu from local mirror (under web root .../apt/mirror/)
Types: deb deb-src
URIs: ${MIRROR_APT_URI}
Suites: noble-proposed
Components: main restricted universe multiverse
Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg
EOF
}

_hook_mirror_write_fwupd_local_remote() {
 mkdir -p /etc/fwupd/remotes.d
 cat > /etc/fwupd/remotes.d/local-lvfs-mirror.conf <<EOF
[fwupd Remote]
```

(continues on next page)

(continued from previous page)

```
Enabled=true
Type=download
Title=Local LVFS Mirror
MetadataURI=${MIRROR_FW_BASE}/${LVFS_METADATA_NAME}
FirmwareBaseURI=${MIRROR_FW_BASE}
EOF
}

_hook_mirror_disable_public_lvfs() {
 fwupdmgr disable-remote lvfs 2>/dev/null || {
   [ -f /etc/fwupd/remotes.d/lvfs.conf ] && \
     mv /etc/fwupd/remotes.d/lvfs.conf \
       /etc/fwupd/remotes.d/lvfs.conf.disabled
 }
}

_hook_mirror_apply_local_mirror() {
 _hook_mirror_archive_stock_sources
 _hook_mirror_write_deb822_sources
 _hook_mirror_write_fwupd_local_remote
 _hook_mirror_disable_public_lvfs
}

# Retry apt update to tolerate transient lock contention (e.g. another apt
↪process).
# 600 cycles, 1 second each.
_hook_apt_update_with_retries() {
 _attempt=1
 _max_attempts=600
 while [ "$_attempt" -le "$_max_attempts" ]; do
   if apt-get update -o Acquire::Languages=none || apt-get update; then
     return 0
   fi
   if [ "$_attempt" -lt "$_max_attempts" ]; then
     echo "apt update failed (attempt ${_attempt}/${_max_attempts});" \
       "retrying in 1s..."
     sleep 1
   fi
   _attempt=$((_attempt + 1))
 done
 echo "apt update failed after ${_max_attempts} attempts"
 return 1
}

# Succeed if either update works; exit 255 only when both fail.
_hook_apt_update_initial() {
 if _hook_apt_update_with_retries; then
   return 0
 fi
 exit 255
}
```

(continues on next page)

(continued from previous page)

```
# ------------------------------------------------------------------------
# OEMDATA: local .deb drop, optional apt repo, firmware USB, optional LVFS URL
# ------------------------------------------------------------------------
_hook_oem_install_debs_from_usb() {
 echo "Checking for debs in USB OEMDATA partition..."
 if [ -d "$OEM_MNT/debs" ] && ls "$OEM_MNT/debs"/*.deb >/dev/null 2>&1; then
   echo "Installing debs from USB OEMDATA partition..."
   dpkg -i "$OEM_MNT/debs"/*.deb || true
   apt-get install -f -y
 fi
}

_hook_oem_install_from_local_repo() {
 echo "Checking for local APT repo..."
 if [ ! -f "$OEM_MNT/apt-repo.url" ]; then
   echo "apt-repo.url not found"
   return 0
 fi
 repo_url=$(sed -n '1s/[[:space:]]*//p' "$OEM_MNT/apt-repo.url")
 if [ -z "$repo_url" ]; then
   echo "No local APT repo URL found"
   return 0
 fi
 echo "Adding local APT repo: $repo_url"
 printf 'deb [trusted=yes] %s ./\n' "$repo_url" > \
   /etc/apt/sources.list.d/oem-local.list
 repo_host=$(echo "$repo_url" | sed -n 's|.*://\([^:/]*\).*|\1|p')
 if [ -n "$repo_host" ]; then
   rm -f /var/lib/apt/lists/partial/*"$repo_host"* \
     /var/lib/apt/lists/*"$repo_host"* 2>/dev/null || true
 fi
 _hook_apt_update_with_retries || true

 if [ ! -f "$OEM_MNT/apt-packages.txt" ]; then
   echo "apt-packages.txt not found; apt upgrade using OEM local repo only" \
     "(single-source apt)"
   apt upgrade -y \
     -o Dir::Etc::sourcelist="/etc/apt/sources.list.d/oem-local.list" \
     -o APT::Architecture="$(dpkg --print-architecture)" \
     || true
   return 0
 fi
 pkgs=$(grep -v '^[#;]' "$OEM_MNT/apt-packages.txt" | while read -r line; do
   line="${line%%[[:space:]]*}"
   [ -z "$line" ] && continue
   case "$line" in *\.deb)
     line="${line%.deb}"; line="${line%_*}"; line="${line%_*}"
   esac
   echo "$line"
 done | tr '\n' ' ')
 if [ -z "$pkgs" ]; then
   return 0
```

(continues on next page)

(continued from previous page)

```
 fi
 echo "Installing packages from local repo: $pkgs"
 if ! apt-get install -y $pkgs; then
   echo "Fallback: downloading .deb and installing with dpkg..."
   base="${repo_url%/}"
   for pkg in $pkgs; do
     pkg_file=$(
       _hook_http_get "$base/Packages.gz" | zcat 2>/dev/null | awk -v pkg="
↪$pkg" '
/^Package: /{name=$2}
/^Filename: /{if(name==pkg){print $2; exit}}
/^$/{name=""}
'
     )
     pkg_file="${pkg_file#./}"
     [ -z "$pkg_file" ] && continue
     tmp_deb="/tmp/$(basename "$pkg_file")"
     if _hook_http_get "$base/$pkg_file" > "$tmp_deb" 2>/dev/null && \
       [ -s "$tmp_deb" ]; then
       dpkg -i "$tmp_deb" && echo "PASS: $pkg (dpkg)" || true
     else
       echo "FAIL: could not download $pkg"
     fi
     rm -f "$tmp_deb"
   done
   apt-get install -f -y 2>/dev/null || true
 fi
}

_hook_oem_install_firmware_usb() {
 echo "Checking for firmware in USB OEMDATA partition..."
 if [ ! -d "$OEM_MNT/firmware" ]; then
   return 0
 fi
 echo "Installing firmware from USB OEMDATA partition..."
 find "$OEM_MNT/firmware" -maxdepth 1 -type f \
   \( -name '*.cab' -o -name '*.cap' \) | while read -r f; do
   name=$(basename "$f")
   echo "Installing firmware: $name"
   if fwupdmgr install --allow-reinstall "$f"; then
     echo "PASS: $name"
   else
     echo "FAIL: $name"
   fi
 done
}

# Supports (1) full LVFS mirror: URL points to dir with firmware.xml.gz;
# (2) directory of .cab/.cap only.
_hook_oem_lvfs_mirror_from_url() {
 echo "Checking for LVFS mirror URL..."
 if [ ! -f "$OEM_MNT/lvfs-mirror.url" ]; then
```

(continues on next page)

(continued from previous page)

```
    return 0
  fi
  lvfs_base=$(sed -n '1s/[[:space:]]*//p' "$OEM_MNT/lvfs-mirror.url")
  lvfs_base="${lvfs_base%/}/"
  if [ -z "$lvfs_base" ]; then
    return 0
  fi
  lvfs_meta_url="${lvfs_base}firmware.xml.gz"
  curl_meta_code=$(curl -sI -o /dev/null -w '%{http_code}' \
    "$lvfs_meta_url" 2>/dev/null)
  if ( wget -q --spider "$lvfs_meta_url" 2>/dev/null ) || \
    [ "$curl_meta_code" = "200" ]; then
    echo "Adding LVFS mirror (metadata): $lvfs_base"
    mkdir -p /etc/fwupd/remotes.d
    cat > /etc/fwupd/remotes.d/oem-lvfs-mirror.conf << EOF
[fwupd Remote]
Title=OEM LVFS Mirror
MetadataURI=${lvfs_base}firmware.xml.gz
FirmwareBaseURI=$lvfs_base
Enabled=true
EOF
    echo "Refreshing fwupd and upgrading firmware from mirror..."
    fwupdmgr refresh --force || fwupdmgr refresh
    fwupdmgr update || true
  else
    echo "No firmware.xml.gz at $lvfs_base; treating as directory of .cab/.cap.
↪.."
    _hook_http_get "$lvfs_base" | grep -oE 'href="[^"]*\.(cab|cap)"' | \
      sed 's/href="//;s/"$//' | while read -r f; do
      [ -z "$f" ] && continue
      tmp_f="/tmp/$(basename "$f")"
      if _hook_http_get "$lvfs_base$f" > "$tmp_f" 2>/dev/null && [ -s "$tmp_f"
↪]; then
        echo "Installing firmware from mirror: $f"
        fwupdmgr install --allow-reinstall "$tmp_f" && \
          echo "PASS: $f" || echo "FAIL: $f"
      fi
      rm -f "$tmp_f"
    done
  fi
}

# ------------------------------------------------------------------------------
# Final pass: apt upgrade + fwupd loop; sets HOOK_APT_UPDATED,
# HOOK_FW_UPDATED, HOOK_FW_FAILED
# ------------------------------------------------------------------------------
_hook_final_apt_upgrade() {
  echo "apt update with retry (600x, 1s)"
  if _hook_apt_update_with_retries; then
    :
  else
    echo "apt update failed"
```

(continues on next page)

(continued from previous page)

```
  exit 255
 fi

 echo "apt -s upgrade | grep -q '^[[:space:]]*Inst '"
 if apt -s upgrade | grep -q '^[[:space:]]*Inst '; then
   DEBIAN_FRONTEND=noninteractive apt -y \
     -o Dpkg::Options::=--force-confold upgrade || exit 255
   HOOK_APT_UPDATED=1
 fi
}

_hook_fwupdmgr_refresh_and_count() {
 echo "fwupdmgr refresh --force"
 fwupdmgr refresh --force

 HOOK_FW_APPLICABLE=""
 if command -v jq >/dev/null 2>&1; then
   _hook_jq_fw_count='(if type == "object" and (.Devices | type) == "array" '
   _hook_jq_fw_count="${_hook_jq_fw_count}then .Devices "
   _hook_jq_fw_count="${_hook_jq_fw_count}"'elif type == "array" then . else '
   _hook_jq_fw_count="${_hook_jq_fw_count}[] end) | "
   _hook_jq_fw_count="${_hook_jq_fw_count}"'[.[] | select((.Releases // []) |
↪'
   _hook_jq_fw_count="${_hook_jq_fw_count}"'length > 0)] | length'
   HOOK_FW_APPLICABLE=$(
     fwupdmgr get-upgrades --json 2>/dev/null | jq -r "$_hook_jq_fw_count" \
       2>/dev/null || echo ""
   )
 fi
 HOOK_FW_APPLICABLE=$(printf '%s' "$HOOK_FW_APPLICABLE" | tr -d '\r\n\t ')
 case "$HOOK_FW_APPLICABLE" in
   ''|*[!0-9]*) HOOK_FW_APPLICABLE="?" ;;
 esac
 echo "fwupdmgr: applicable firmware devices (Releases>0): ${HOOK_FW_
↪APPLICABLE:-?}"
}

_hook_fwupdmgr_upgrade_loop() {
 set +e
 case "$HOOK_FW_APPLICABLE" in
   [1-9]|[1-9][0-9]*)
     HOOK_FW_ITER=0
     while [ "$HOOK_FW_ITER" -lt 5 ]; do
       HOOK_FW_ITER=$((HOOK_FW_ITER + 1))
       HOOK_FW_OFFLINE_CAN_BREAK=0
       HOOK_FW_IMMEDIATE_CAN_BREAK=0
       echo "fwupdmgr upgrade -y --offline"
       fwupdmgr upgrade -y --offline
       HOOK_FW_R1=$?
       echo "fwupdmgr upgrade -y --no-reboot-check"
       fwupdmgr upgrade -y --no-reboot-check
       HOOK_FW_R2=$?
```

(continues on next page)

(continued from previous page)

```
      case $HOOK_FW_R1 in
        0) HOOK_FW_UPDATED=1; HOOK_FW_OFFLINE_CAN_BREAK=1 ;;
        2) HOOK_FW_OFFLINE_CAN_BREAK=1 ;;
        *) echo "fwupdmgr upgrade -y --offline failed (exit $HOOK_FW_R1)"; \
           HOOK_FW_FAILED=1 ;;
      esac
      case $HOOK_FW_R2 in
        0) HOOK_FW_UPDATED=1; HOOK_FW_IMMEDIATE_CAN_BREAK=1 ;;
        2) HOOK_FW_IMMEDIATE_CAN_BREAK=1 ;;
        *) echo "fwupdmgr upgrade -y --no-reboot-check failed (exit $HOOK_FW_
↪R2)"; \
           HOOK_FW_FAILED=1 ;;
      esac
      if [ "$HOOK_FW_FAILED" -eq 1 ]; then
        break
      fi
      if [ "$HOOK_FW_OFFLINE_CAN_BREAK" -eq 1 ] && \
        [ "$HOOK_FW_IMMEDIATE_CAN_BREAK" -eq 1 ]; then
        echo "fwupdmgr upgrade -y --offline and fwupdmgr upgrade -y" \
          "--no-reboot-check both finished OK"
        break
      fi
    done
    ;;
 esac
 set -e
}

_hook_restore_stock_apt_and_lvfs() {
 echo "Restoring stock apt sources and re-enabling public LVFS after" \
   "mirror-based updates"
 if [ -d /etc/apt/sources.list.d.org ]; then
   if [ -d /etc/apt/sources.list.d.cldnt ]; then
     rm -rf /etc/apt/sources.list.d.cldnt
   fi
   if [ -d /etc/apt/sources.list.d ]; then
     mv /etc/apt/sources.list.d /etc/apt/sources.list.d.cldnt
   fi
   mv /etc/apt/sources.list.d.org /etc/apt/sources.list.d
 fi
 if [ -f /etc/fwupd/remotes.d/lvfs.conf.disabled ]; then
   mv /etc/fwupd/remotes.d/lvfs.conf.disabled /etc/fwupd/remotes.d/lvfs.conf
 fi
 # Non-interactive: enable-remote otherwise blocks on "Enable new remote?"
# (no TTY under cloud-init).
# The LVFS disclaimer box goes to stdout; cloud-init captures runcmd output into
# cloud-init-provisioning.log.
 fwupdmgr -y --no-remote-check enable-remote lvfs >/dev/null 2>&1 || true
}

_hook_exit_with_status() {
 if [ "$HOOK_FW_FAILED" -eq 1 ]; then
```

(continues on next page)

(continued from previous page)

```
    exit 255
  fi
  if [ "$HOOK_APT_UPDATED" -eq 1 ] || [ "$HOOK_FW_UPDATED" -eq 1 ]; then
    exit 1
  fi
  exit 0
}

# ------------------------------------------------------------------------
# Main
# ------------------------------------------------------------------------
_hook_main() {
  _hook_config_defaults
  _hook_mirror_apply_local_mirror
  _hook_oem_install_debs_from_usb
  _hook_oem_install_from_local_repo
  _hook_oem_install_firmware_usb
  _hook_oem_lvfs_mirror_from_url

  HOOK_APT_UPDATED=0
  HOOK_FW_UPDATED=0
  HOOK_FW_FAILED=0
  _hook_fwupdmgr_refresh_and_count
  _hook_fwupdmgr_upgrade_loop
  _hook_apt_update_initial
  _hook_final_apt_upgrade
  _hook_restore_stock_apt_and_lvfs
  _hook_exit_with_status
}

_hook_main
```

2.9.2.17.2 Cloud-init Configurations

The following examples provide the two required seed files. Place `meta-data` and the applicable `user-data` file in the root of the FAT32 `OEMDATA` partition.

2.9.2.17.2.1 meta-data

The following file sets the cloud-init instance identifier:

```
# instance-id is used by cloud-init as a unique instance identifier.

instance-id: oem-spark-01
```

2.9.2.17.2.2 user-data with User Creation (Skip OOBE)

Use this example when cloud-init creates the user and OOBE must be skipped.

Warning

If you use custom `user-data` to skip OOBE, create `/etc/cloud/cloud-init.disabled` to disable cloud-init on subsequent boots. The following example creates this file automatically.

```
#cloud-config
# Default user(YOUR_USERNAME:YOUR_PASSWORD)

growpart:
  mode: 'off'

no_ssh_fingerprints: true
resize_rootfs: false

# Allow password auth for console and SSH (required for YOUR_USERNAME login)
ssh_pwauth: true

# Create default user YOUR_USERNAME with password YOUR_PASSWORD
# Use hashed password so login works reliably (chpasswd as backup)
# **** REMOVE BOTH USERS AND CHPASSWD IF USER WANT TO RUN OOBE ****
users:
  - name: YOUR_USERNAME
    groups: [sudo]
    shell: /bin/bash
    lock_passwd: false
    # SHA-512 hash for password "YOUR_PASSWORD" (random 16-char salt)
    hashed_passwd: #GET_WITH_THIS_COMMAND <openssl passwd -6 -salt \
      random_16-char_salt "YOUR_PASSWORD">

chpasswd:
  expire: false
  users:
    - name: YOUR_USERNAME
      password: YOUR_PASSWORD
      type: text

runcmd:
  # hostnamectl can fail on first boot if systemd-hostnamed/D-Bus is not ready
  # yet; do not rely on it alone.
  - |
    echo YOUR_USERNAME > /etc/hostname
    hostname YOUR_USERNAME 2>/dev/null || true
    if grep -qE '^127\.0\.1\.1[[:space:]]+YOUR_USERNAME' /etc/hosts 2>/dev/
↪null; then
      :
    elif grep -qE '^127\.0\.1\.1' /etc/hosts 2>/dev/null; then
      sed -i 's/^127\.0\.1\.1.*/127.0.1.1 YOUR_USERNAME/' /etc/hosts
    else
      echo "127.0.1.1 YOUR_USERNAME" >> /etc/hosts
    fi
    hostnamectl set-hostname YOUR_USERNAME 2>/dev/null || true
  - [ sh, -c, 'userdel ubuntu 2>/dev/null || true' ]
```

(continues on next page)

(continued from previous page)

```
  - [ sh, -c, 'rm -rf /home/ubuntu 2>/dev/null || true' ]
  - |
    echo "OEM cloud-init seed ran at $(date -Iseconds)" \
      >> /var/log/oem-cloud-init-seed.log
  - [ chmod, '0644', /var/log/oem-cloud-init-seed.log ]
  - |
    mkdir -p /etc/ssh/sshd_config.d
    printf '%s\n' 'PasswordAuthentication yes' \
      'ChallengeResponseAuthentication no' \
      > /etc/ssh/sshd_config.d/99-oem-password-auth.conf
    systemctl reload sshd 2>/dev/null || true
  # Mount USB data partition (label OEMDATA) and run hook.sh if present
  # (hook installs debs/firmware)
  - |
    OEM_MNT=/mnt/oemdata
    HOOK_RUN=/tmp/oemdata-hook.sh
    NET_HOOK=/tmp/network-hook.sh
    HOOK_EXECUTED=0
    mkdir -p "$OEM_MNT"
    # One trap for all normal completion: umount + disable cloud-init on next
↪boots.
    # Do not use "exit" here: cloud-init shellifies all runcmd items into one
↪/bin/sh
    # script; exit would skip every later runcmd line (e.g. pending-reboot
    # check) before the EXIT trap runs.
    # Do not use "set -e" in this block: if sync/mkdir/touch after the hook
    # fails, the shell would exit before the post-hook runcmd; EXIT would
↪still
    # run _oemdata_exit (cloud-init.disabled) but the pending-reboot log/
↪reboot
    # would never run.
    _oemdata_exit() {
      echo "OEMDATA exit trap: disabling cloud-init and unmounting OEMDATA"
      umount "$OEM_MNT" 2>/dev/null || true
      rmdir "$OEM_MNT" 2>/dev/null || true
      mkdir -p /etc/cloud
      touch /etc/cloud/cloud-init.disabled
    }
    _run_oem_hook() {
      _src="$1"
      _origin="$2"
      cp -f "$_src" "$HOOK_RUN" || return 1
      chmod 700 "$HOOK_RUN"
      echo "Running OEM hook from $HOOK_RUN (source: $_origin)"
      set +e
      export OEM_MNT
      sh "$HOOK_RUN"
      _hook_rc=$?
      rm -f "$HOOK_RUN"
      HOOK_EXECUTED=1
      if [ "$_hook_rc" -eq 1 ]; then
        echo "mirror setup success"
```

(continues on next page)

(continued from previous page)

```
      sync
      mkdir -p /var/lib/oem
      touch /var/lib/oem/oem-hook-pending-reboot
    elif [ "$_hook_rc" -ne 0 ]; then
      echo "OEM hook exited with code $_hook_rc"
    fi
    return 0
  }
  _iface_ipv4() {
    ip -4 -o addr show dev "$1" 2>/dev/null | awk '{print $4}' | \
      cut -d/ -f1 | head -n1
  }
  _download_hook() {
    _url="$1"
    _out="$2"
    if command -v curl >/dev/null 2>&1; then
      curl -fsSL --connect-timeout 5 --max-time 20 "$_url" -o "$_out"
    elif command -v wget >/dev/null 2>&1; then
      wget -q -T 20 -O "$_out" "$_url"
    else
      echo "Neither curl nor wget is available to download network hook."
      return 1
    fi
  }
  trap '_oemdata_exit' EXIT
  if mount -L OEMDATA "$OEM_MNT" 2>/dev/null; then
    echo "OEMDATA partition found, checking for hook.sh"
    if [ -f "$OEM_MNT/hook.sh" ]; then
      _run_oem_hook "$OEM_MNT/hook.sh" "OEMDATA"
    fi
  else
    echo "No OEMDATA partition found, skipping USB OEM hook."
    rmdir "$OEM_MNT" 2>/dev/null || true
  fi
# cloud-init.disabled is created in OEM block EXIT trap above
# (covers no-OEMDATA path too).
# Reboot if OEM hook requested it (hook exit 1). Runs in same shellified
↪script
# after OEM block (no "exit" above).
- |
  echo "OEM post-hook: checking pending-reboot marker"
  if [ -f /var/lib/oem/oem-hook-pending-reboot ]; then
    rm -f /var/lib/oem/oem-hook-pending-reboot
    echo "reboot required (OEM mirror apt/fwupd updates); scheduling reboot
↪" \
      "(+30s so cloud-init can finish modules-final)"
    # EXIT trap may not run before reboot; disable cloud-init and unmount
    # OEMDATA now.
    _oemdata_exit
    sync
    # Immediate reboot races remaining modules-final (e.g. cc_keys_to_
↪console)
```

(continues on next page)

(continued from previous page)

```
      # and can log SystemExit:1.
      # Background sleep + reboot: runcmd exits, cloud-init completes, then
↪reboot
      # (shutdown +m is minute-only).
      ( sleep 30; /sbin/reboot ) </dev/null >/dev/null 2>&1 &
    else
      echo "reboot not required (no OEM pending-reboot marker)"
    fi
```

2.9.2.17.2.3 user-data without User Creation (Keep OOBE)

Use this example when cloud-init must run provisioning without creating a user. The provisioning worker runs from `bootcmd` because `runcmd` does not run when OOBE runs. Provisioning runs while OOBE remains available for the user to create an account. This example sets `preserve_hostname: true` and does not change the hostname. OOBE sets the hostname.

The worker runs separately from cloud-init so OOBE can proceed. It first tries `hook.sh` from `OEMDATA`. If the hook is not present, the worker tries the configured network location. If the hook runs, cloud-init is disabled after the hook exits, regardless of the hook exit code. If no hook runs, the next boot retries the workflow.

Before using this example, replace the example values for `CLOUDINIT_IFACE`, `CLOUDINIT_SERVER_IP` (`192.168.1.1`), and `CLOUDINIT_SERVER_PORT` (`8080`) with values for your installation network.

> **Warning**
>
> The network fallback downloads `hook.sh` over HTTP and runs it as `root`. Use this fallback only on a trusted installation network, and ensure that only authorized administrators can modify the hosted script.

```
#cloud-config
# No cloud-init user/password creation; leave account setup to OOBE / image
↪defaults.

growpart:
  mode: 'off'

no_ssh_fingerprints: true
resize_rootfs: false

# Mandatory on an image that boots into OOBE. dgx-oobe-hostname.service runs
# set-hostname.sh, which rewrites /etc/hostname to spark-<4 hex>, clears
# /etc/machine-id, regenerates the SSH host keys and reboots whenever
# /etc/hostname does not already hold that name. The seed carries no
# local-hostname, so cc_set_hostname falls back to 'localhost' and
# cc_update_hostname re-asserts it on every boot at frequency always. The two
# then overwrite each other indefinitely: cloud-init puts the fallback name
# back, set-hostname.sh sees the mismatch and reboots, and provisioning never
# gets to write the cloud-init.disabled gate. Leaving the hostname alone is
# what breaks the cycle; OOBE owns the hostname on these units.
```

(continues on next page)

(continued from previous page)

```
preserve_hostname: true

# Explicitly create no users (avoids distro default_user creation).
users: []

# The payload cannot run from runcmd on an image that boots into OOBE: runcmd is
# written by modules:config but executed by scripts-user in modules:final, and
# cloud-final.service is ordered After=multi-user.target, which stays unreached
# while the OOBE session holds plymouth-quit-wait.service. bootcmd runs in the
# init stages, which always complete, so the payload is launched from there.
#
# bootcmd runs inside cloud-init-local.service: DefaultDependencies=no,
# Before=sysinit.target, i.e. before basic userspace, with systemd's initial
↪boot
# transaction still in flight. Nothing here may talk to systemd: a daemon-reload
# at this point re-runs every generator and rebuilds the unit graph mid-boot,
# which leaves the boot unable to progress until the SBSA watchdog resets the
# machine. The payload is therefore a plain detached process, not a unit.
# cloud-init-local.service is Type=oneshot with RemainAfterExit=yes, so systemd
# leaves its cgroup alone and the child outlives the stage.
bootcmd:
  - |
    # /etc/cloud/cloud-init.disabled is the whole provisioning gate: the
↪generator
    # honours it before any of this runs, so a boot that gets here is a boot
↪that
    # should provision. The marker below is a per-boot guard only, so that
↪the two
    # bootcmd invocations per boot (init-local and init) start one worker,
↪not two.
    if [ ! -e /run/oemdata-provision.armed ]; then
    # The heredoc must stay at this indentation: YAML strips the block's base
    # indent, and a heredoc terminator cannot carry leading whitespace.
    cat > /run/oemdata-provision.sh <<'OEM_WORKER'
    #!/bin/sh
    # Detached from cloud-init and from systemd: nothing here can block the
↪boot
    # transaction. All output goes to the seed log, there being no journal
↪unit.
    OEM_MNT=/mnt/oemdata
    HOOK_RUN=/tmp/oemdata-hook.sh
    NET_HOOK=/tmp/network-hook.sh
    HOOK_EXECUTED=0
    SEED_LOG=/var/log/cloud-init-output.log
    exec >>"$SEED_LOG" 2>&1
    chmod 0644 "$SEED_LOG"
    echo "OEM cloud-init seed ran at $(date -Iseconds)"

    # Wait for the network the hook needs. This is not a wait on OOBE:
    # dgx-oobe.service is up on every boot long before the user touches the
    # screen, so gating on it would gate on nothing.
    _waited=0
```

(continues on next page)

(continued from previous page)

```
    while [ "$_waited" -lt 60 ]; do
      [ -n "$(ip -4 route show default 2>/dev/null)" ] && break
      sleep 2
      _waited=$((_waited + 2))
    done
    echo "default route after ${_waited}s: $(ip -4 route show default 2>/dev/
↪null | head -n1)"

    # Do not set the hostname here. OOBE owns it: dgx-oobe-hostname.service
↪runs
    # set-hostname.sh, which requires /etc/hostname to be spark-<4 hex> and,
↪on
    # any mismatch, rewrites it, clears /etc/machine-id, regenerates the SSH
↪host
    # keys and reboots. Writing a name of our own made that mismatch true on
    # every boot, so the unit rebooted before provisioning could write the
    # cloud-init.disabled gate, and the next boot started the same race again.
    # That is the reboot loop seen when a unit is flashed and first-booted
↪with
    # the OEMDATA stick still attached.

    mkdir -p "$OEM_MNT"
    _oemdata_exit() {
      umount "$OEM_MNT" 2>/dev/null || true
      rmdir "$OEM_MNT" 2>/dev/null || true
      if [ "$HOOK_EXECUTED" -eq 1 ]; then
        echo "OEMDATA exit trap: disabling cloud-init and unmounting OEMDATA"
        mkdir -p /etc/cloud
        touch /etc/cloud/cloud-init.disabled
      else
        echo "OEMDATA exit trap: unmounted, but not disabling cloud-init
↪(hook_executed=$HOOK_EXECUTED); next boot will retry"
      fi
    }
    _run_oem_hook() {
      _src="$1"
      _origin="$2"
      cp -f "$_src" "$HOOK_RUN" || return 1
      chmod 700 "$HOOK_RUN"
      echo "Running OEM hook from $HOOK_RUN (source: $_origin)"
      export OEM_MNT
      sh "$HOOK_RUN"
      _hook_rc=$?
      rm -f "$HOOK_RUN"
      HOOK_EXECUTED=1
      if [ "$_hook_rc" -eq 1 ]; then
        echo "mirror setup success"
        sync
        mkdir -p /var/lib/oem
        touch /var/lib/oem/oem-hook-pending-reboot
      elif [ "$_hook_rc" -ne 0 ]; then
        echo "OEM hook exited with code $_hook_rc"
```

(continues on next page)

(continued from previous page)

```
    fi
    return 0
  }
  _iface_ipv4() {
    ip -4 -o addr show dev "$1" 2>/dev/null | awk '{print $4}' | cut -d/ -
↪f1 | head -n1
  }
  _download_hook() {
    _url="$1"
    _out="$2"
    if command -v curl >/dev/null 2>&1; then
      curl -fsSL --connect-timeout 5 --max-time 20 "$_url" -o "$_out"
    elif command -v wget >/dev/null 2>&1; then
      wget -q -T 20 -O "$_out" "$_url"
    else
      echo "Neither curl nor wget is available to download network hook."
      return 1
    fi
  }
  trap '_oemdata_exit' EXIT
  if mount -L OEMDATA "$OEM_MNT" 2>/dev/null; then
    echo "OEMDATA partition found, checking for hook.sh"
    if [ -f "$OEM_MNT/hook.sh" ]; then
      _run_oem_hook "$OEM_MNT/hook.sh" "OEMDATA"
    fi
  else
    echo "No OEMDATA partition found, skipping USB OEM hook."
    rmdir "$OEM_MNT" 2>/dev/null || true
  fi
  if [ "$HOOK_EXECUTED" -eq 0 ]; then
    CLOUDINIT_IFACE="${CLOUDINIT_IFACE:-enP7s7}"
    CLOUDINIT_SERVER_IP="${CLOUDINIT_SERVER_IP:-192.168.1.1}"
    CLOUDINIT_SERVER_PORT="${CLOUDINIT_SERVER_PORT:-8080}"
    HOOK_URL="http://${CLOUDINIT_SERVER_IP}:${CLOUDINIT_SERVER_PORT}/hook.sh
↪"
    echo "USB hook not executed. Trying network hook on ${CLOUDINIT_IFACE}
↪from ${HOOK_URL}"
    _attempt=1
    while [ "$_attempt" -le 2 ]; do
      _ip="$(_iface_ipv4 "$CLOUDINIT_IFACE")"
      if [ -z "$_ip" ]; then
        echo "No DHCP IPv4 on ${CLOUDINIT_IFACE} (attempt ${_attempt}/2),
↪requesting lease"
        if command -v dhclient >/dev/null 2>&1; then
          dhclient -4 -1 "$CLOUDINIT_IFACE" 2>/dev/null || true
        else
          echo "dhclient not found, cannot request DHCP lease."
        fi
        _ip="$(_iface_ipv4 "$CLOUDINIT_IFACE")"
      fi
      if [ -n "$_ip" ]; then
        echo "${CLOUDINIT_IFACE} has IPv4 ${_ip}; downloading ${HOOK_URL}"
```

(continues on next page)

(continued from previous page)

```
          if _download_hook "$HOOK_URL" "$NET_HOOK"; then
            _run_oem_hook "$NET_HOOK" "$HOOK_URL"
            rm -f "$NET_HOOK"
            break
          else
            echo "Failed to download network hook from ${HOOK_URL}"
          fi
        fi
        if [ "$_attempt" -lt 2 ]; then
          echo "Waiting 10s before DHCP/download retry."
          sleep 10
        fi
        _attempt=$((_attempt + 1))
      done
    fi
    echo "OEM post-hook: checking pending-reboot marker"
    if [ -f /var/lib/oem/oem-hook-pending-reboot ]; then
      rm -f /var/lib/oem/oem-hook-pending-reboot
      echo "reboot required (OEM mirror apt/fwupd updates); scheduling reboot
↪in 30s"
      # EXIT trap may not run before reboot; disable cloud-init and unmount
↪OEMDATA now.
      _oemdata_exit
      trap - EXIT
      sync
      ( sleep 30; /sbin/reboot ) </dev/null >/dev/null 2>&1 &
    else
      echo "reboot not required (no OEM pending-reboot marker)"
    fi
    OEM_WORKER
    # Fully detached: no controlling terminal, and cloud-init's stdout pipe
↪is not
    # inherited, which would otherwise make cloud-init wait for the hook.
    setsid /bin/sh /run/oemdata-provision.sh </dev/null >/dev/null 2>&1 &
    touch /run/oemdata-provision.armed
    fi
```

2.9.2.18 Validation Scenarios and Feedback Questions

The validation scenarios in this section align with *Table 1* and the procedures through *Reference: Scripts and cloud-init*. Complete the scenario that matches your deployment. For log-based and post-installation verification, refer to *Verify the Customization and Installation Outcomes*.

How Scenarios Map to This Document

The following table maps each validation scenario to the related installation patterns and sections in this topic:

Scenario	Related Table 1 patterns	Where to Work
Air-gapped USB installation	USB-hosted packages and firmware or local server sources through `OEMDATA` and `hook.sh`	*USB Partitioning and the OEMDATA Layout*; *Host a Minimal APT Repository and Firmware Tree*; *On the DGX Spark Client: hook.sh and OEMDATA Files*; *First Boot: OEMDATA hook.sh and cloud-init Seed*
Local repository and application updates	Curated or mirrored APT layout; can extend beyond first-boot automation	*Host a Minimal APT Repository and Firmware Tree*; *On the DGX Spark Client: hook.sh and OEMDATA Files*; *After Mirror- or USB-Driven Updates*

2.9.2.18.1 Validation Scenarios

Complete each scenario that matches your deployment. Use the checklist in each scenario during deployment.

1. **Air-gapped installation using USB (`OEMDATA`, optional local mirror):** Perform provisioning using Debian packages and firmware supplied from `OEMDATA` and, where applicable, your network mirror.

   Prepare the USB layout per *USB Partitioning and the OEMDATA Layout*. Host packages and firmware using *Host a Minimal APT Repository and Firmware Tree*, then configure the client with `hook.sh` and optional URL files as described in *On the DGX Spark Client: hook.sh and OEMDATA Files*.

   Checklist:

   - Wipe or prepare the USB drive if your process requires it. Reflashing can replace the entire device.
   - Obtain Debian packages and firmware that your process permits on the disconnected network:
     - Use the APT or package-acquisition tools that your OEM or NVIDIA program supplies.
     - Use the program manifest or equivalent bill of materials to determine which software and firmware to stage and which approved sources to use.
     - Populate `OEMDATA` and any server tree using the documented layout and URL files (`debs/`, `firmware/`, `apt-repo.url`, and `lvfs-mirror.url` as applicable).
   - If the system runs a release earlier than the October 2026 Release, reflash it with an October 2026 Release or later recovery image.
   - Power off the system, attach `OEMDATA`, then power on.
   - Verify packages, firmware, and customizations using *After Mirror- or USB-Driven Updates*.

2. **Local repository and application updates:** Use your standard IT administration tools to host a local APT repository, then distribute application packages. Exclude firmware, kernel, and driver components unless your policy permits them.

   The minimal repository workflow shows an HTTP-served package tree. The client workflow describes how to configure the `OEMDATA` URL files. Record any additional steps required by your application-update delivery mechanism in your runbook.

   Checklist:

- Configure the applicable repository URL files to use your local mirror URLs.
- Deploy the updated `user-data` and `meta-data` to the device according to your update process.
- Publish the approved application packages to the local repository.
- Verify that the device receives the expected package updates.

By accessing the instructions for custom installation and scaled deployment, the Customer acknowledges that this workflow bypasses individual end-user license prompts. Use of these tools is governed by the *NVIDIA Software License Agreement* (which includes the NVIDIA CUDA EULA). By proceeding, the Administrator represents they have the authority to bind the Customer to these terms and conditions.

2.10. OS and Component Update Guide

This section provides guidance for updating the operating system, software components, and firmware on your DGX Spark. The DGX Spark runs on NVIDIA DGX OS, which is an Ubuntu-based Linux distribution optimized for AI workloads.

Note

Founders Edition Only: The update information in this guide applies only to the DGX Spark Founders Edition. Devices from other manufacturers might have different update procedures.

2.10.1. Ubuntu Support

The Ubuntu Pro OS license in DGX Spark includes 10-year OS support from Canonical.

2.10.2. Update Methods

We strongly recommend using the DGX Dashboard for all system updates on your |spark|. The dashboard ensures your system stays up to date with the latest NVIDIA-optimized components and configurations. While standard Ubuntu package management methods will work, the dashboard provides the most reliable and tested update path for your DGX system.

2.10.3. Using DGX Dashboard for Updates

The DGX Dashboard is the **primary and recommended** way to perform system updates on your DGX Spark. It provides a centralized interface for:

- Viewing available system updates
- Installing security patches and system updates
- Managing NVIDIA driver updates
- Managing firmware updates
- Monitoring update status and progress

For detailed information about using the DGX Dashboard, including how to access it and perform updates, see *DGX Dashboard*.

Note

Where system administrators manage updates, an administrator can disable the dashboard update interface so that users cannot initiate updates. Refer to *Administrator Policy Settings* for more information.

Warning

Before performing any system or firmware updates:

- Ensure your system is connected to a stable power source
- Close all running applications and save your work
- Have a recovery plan in place. Refer to *System Repair* to repair the installed system, or *System Recovery* to restore the factory image.
- Schedule updates during maintenance windows when possible

2.10.4. Manual System Updates

Note: While manual updates using standard Ubuntu package management commands will work, we strongly recommend using the DGX Dashboard for all updates to ensure optimal system performance and compatibility.

For advanced users or when the DGX Dashboard is not available, you can perform system updates manually using the following steps:

1. Open a remote or local terminal on the DGX Spark device.
2. Run the following commands:

```
sudo apt update
sudo apt dist-upgrade
sudo fwupdmgr refresh
sudo fwupdmgr upgrade
sudo reboot
```

These commands will:

- Update the package lists and upgrade all installed packages (including OS components and drivers)
- Refresh the firmware metadata and upgrade all firmware components
- Reboot the system to apply updates

2.10.5. Update Best Practices

- **Use the DGX Dashboard**: Always prefer the DGX Dashboard for system updates to ensure compatibility and optimal performance

- **Regular updates**: Check for updates regularly, especially security patches
- **Backup before major updates**: Always backup critical data before major system changes
- **Stable power**: Ensure your system has a stable power supply during updates
- **Maintenance windows**: Schedule updates during planned maintenance windows when possible

2.11. System Repair

If your DGX Spark is unable to boot, there are two recommended methods that you can use to recover your device.

Before you begin, choose one of the following recovery methods based on whether your system can reach the GRUB boot menu and boot an earlier kernel.

1. **Previously installed kernel:** If you can access GRUB and your system has an earlier kernel installed, follow the steps below to boot with that kernel. Use this option if a recent kernel update prevents your system from starting normally but an earlier kernel still works.
2. **Bootable USB:** If you cannot access GRUB or boot any installed kernel, use another computer to create a bootable USB drive. Boot the affected device from the USB to access a live environment, where you can mount your drive, repair the installed system, or recover files.

> **Note**
>
> **Founders Edition and OEM systems**: This procedure applies to the DGX Spark Founders Edition and to OEM DGX Spark systems. It does not erase the internal SSD.

2.11.1. Previously Installed Kernel

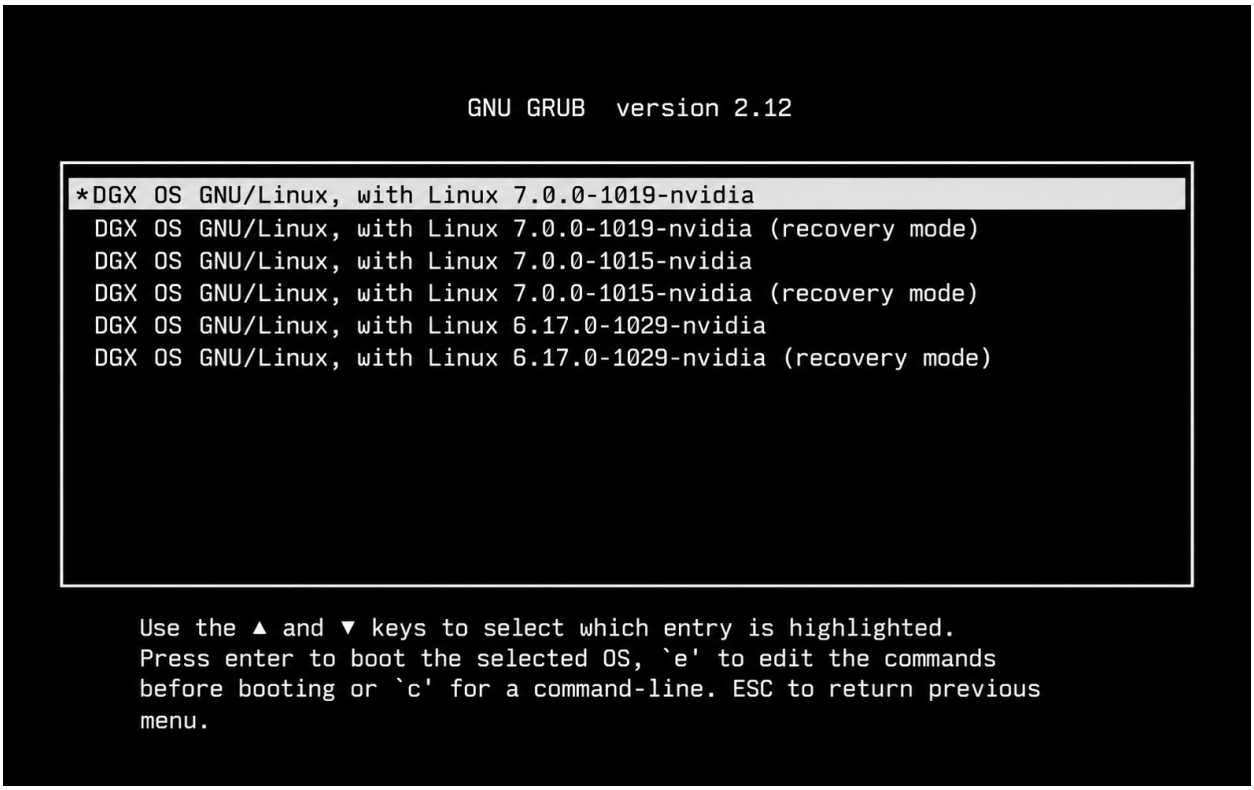
2.11.1.1 Boot from the GRUB Menu

If you have previously installed kernels available on your system, you may be able to use them to boot your DGX Spark. First power off your device, then reboot while holding `Shift` and `Insert` at the same time.

This should bring you to the following GRUB menu:



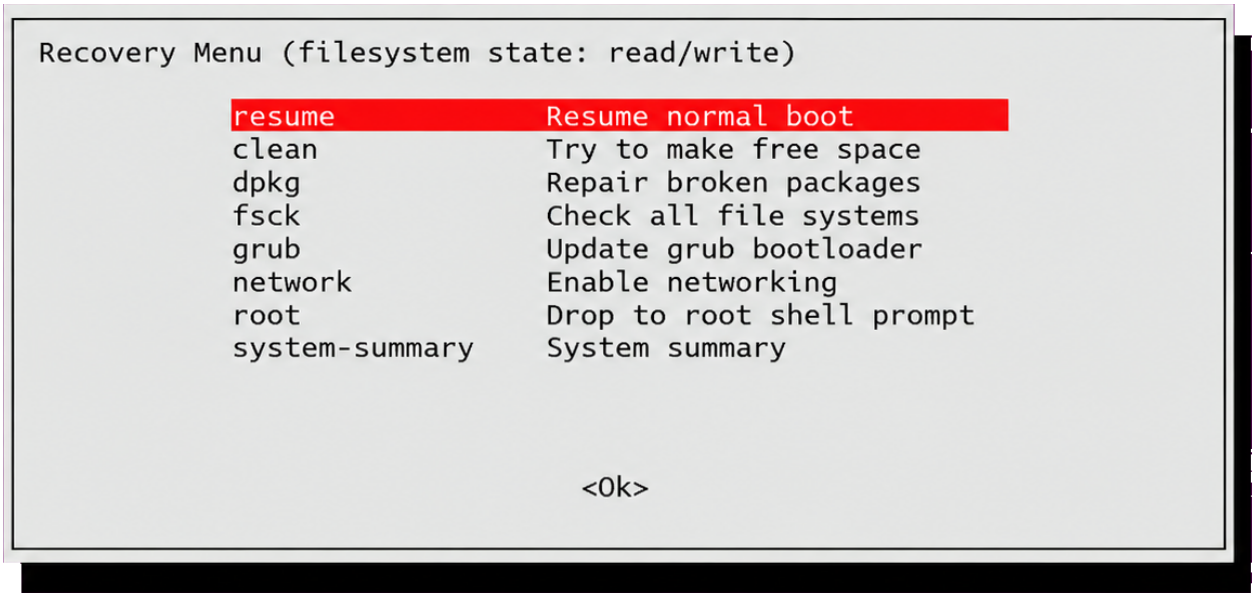
Select “Advanced options for DGX OS GNU/Linux”, which should bring you to the available kernel options:



Select the kernel version you want to boot. Each installed kernel version has a normal boot entry and

a recovery mode entry.

Recovery mode uses the same kernel but starts a limited maintenance environment with the options as shown below. For advanced users of Ubuntu, the recovery mode kernel has an option to get you into a root shell for your device.



2.11.1.2 Common Scenarios for the Booted Kernel

If you booted using a recovery kernel, follow the instructions below for your specific issue. Execute commands directly with root privileges without running any mount, chroot, or unmount steps.

2.11.1.2.1 Scenario 1: Complete a Package Upgrade

This will help if you ran `apt upgrade` instead of `apt dist-upgrade` or need to update a faulty kernel.

1. Update your configuration and complete the upgrade:

```
dpkg --configure -a
# Fix apt upgrade error
apt update
apt -f install
apt dist-upgrade
```

2. Reboot your device:

```
reboot
```

2.11.1.2.2 Scenario 2: Rebuild initramfs

If you have an incomplete or missing initramfs, you can run the following to complete a half-finished upgrade:

1. Rebuild the initramfs:

```
dpkg --configure -a
# Run the following command if you have an interrupted initramfs upgrade
update-initramfs -u -k all
# If you are missing your initramfs entirely, run 'update-initramfs -c -k
 ↪all' instead
update-grub
```

2. Reboot your device:

```
reboot
```

2.11.1.2.3 Scenario 3: Install signed NVIDIA driver packages

Use this procedure if your NVIDIA GPU driver is missing or Secure Boot prevents its kernel modules from loading. The following steps remove leftover modules and install compatible, signed NVIDIA driver packages.

1. Set KVER to the installed kernel version you want to repair and boot:

```
KVER=<kernel version>
```

2. Remove leftover modules from a previous installation.

   If you previously installed the NVIDIA driver with a `.run` installer, run:

```
nvidia-uninstall
```

   If you installed the driver only through APT, skip that command.

   If a previous manual installation left NVIDIA modules behind, remove files that are not owned by a Debian package. Confirm that KVER is set to the installed kernel you intend to repair before continuing:

```
find "/lib/modules/${KVER:?Set KVER first}" \
-name 'nvidia*.ko*' \
-exec sh -c 'dpkg -S "$1" >/dev/null 2>&1 || rm -v "$1"' _ {} \;
depmod -a "$KVER"
```

3. Find and install the signed packages:

```
apt update
apt-cache search linux-modules-nvidia | grep -F -- "$KVER"
apt-cache search nvidia-driver | grep -i open
# Set the corresponding modules and drivers to be the ones for your
 ↪selected kernel version
apt install <signed linux-modules-nvidia package> <nvidia-driver package>
```

> **Warning**
>
> Do not install `nvidia-dkms-*` modules as they are unsigned. The DGX Spark has Secure Boot enabled, so these modules will fail to load unless they are signed with a trusted key.

4. Rebuild, verify, and reboot:

```
update-initramfs -u -k $KVER
modinfo -k $KVER -F signer nvidia # must be non-empty
reboot
```

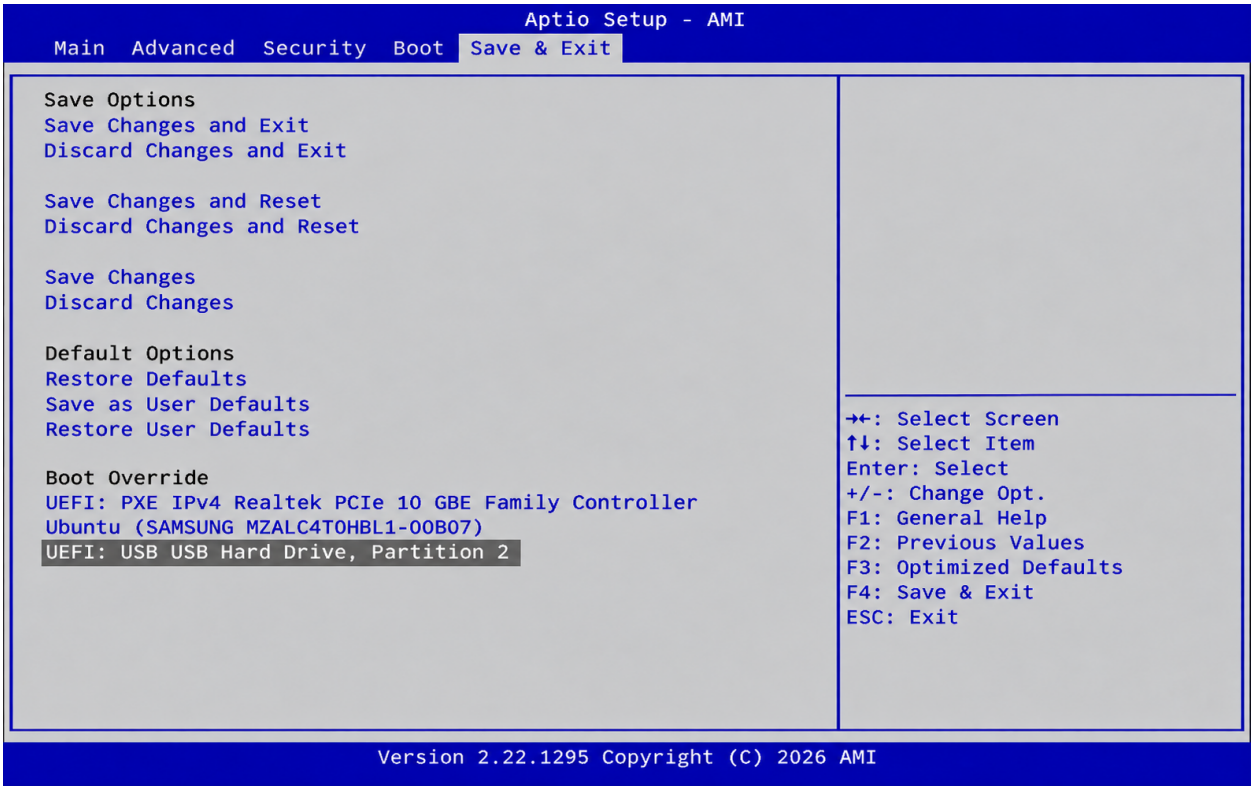
2.11.2. Bootable USB

2.11.2.1 Download the Ubuntu Desktop ISO

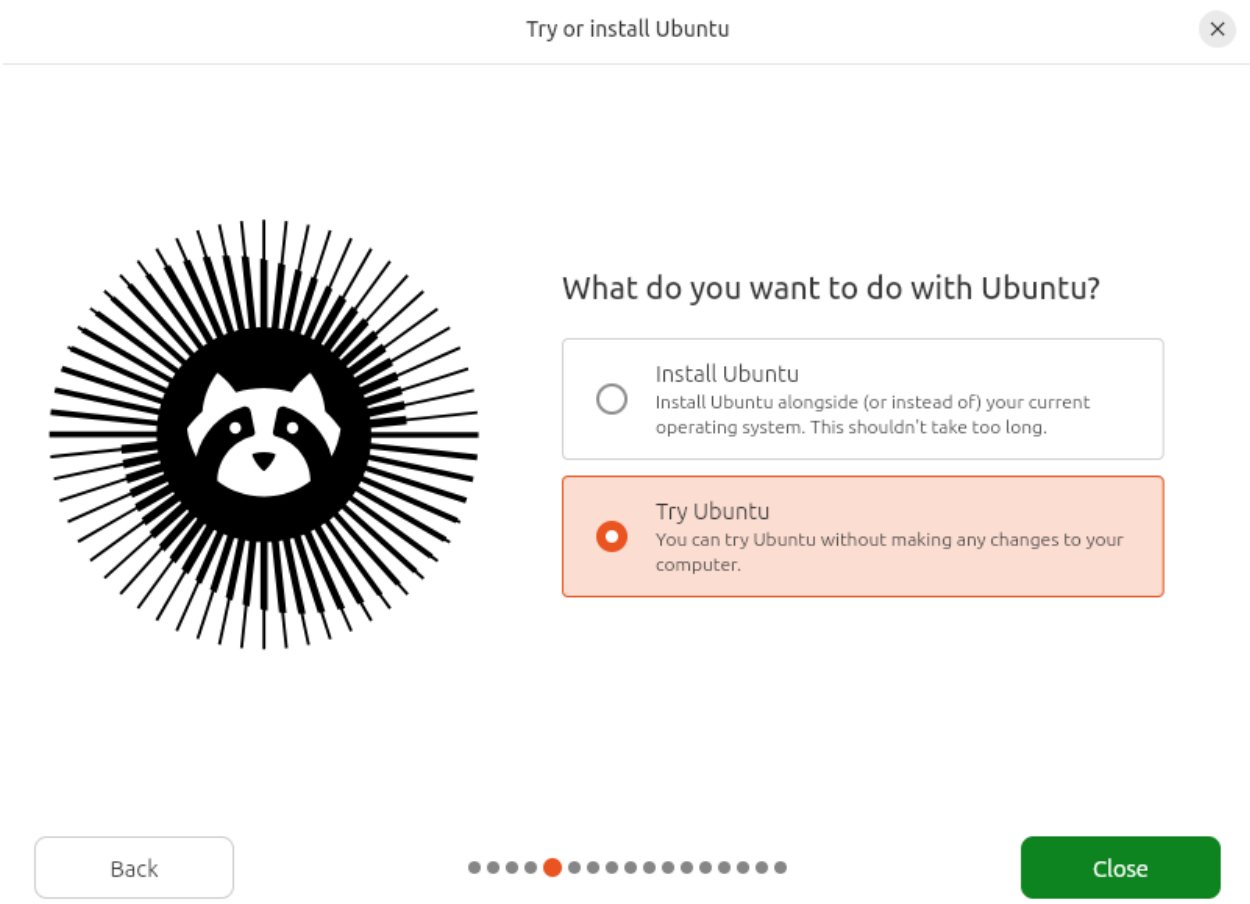
- Ubuntu's default installer supports both full OS installation and live USB booting.
- Download the ARM 64-bit architecture version of the installer from the Ubuntu Desktop download page. The current version is Ubuntu 26.04.1 LTS "ARM 64-bit architecture".
- Create a bootable USB flash drive with an image writer.

2.11.2.2 Boot into Ubuntu

Insert your USB drive into your DGX Spark, then hold Esc while it boots. The UEFI menu will appear with a new USB entry under boot options. Select the USB entry to proceed; your system may reboot, but it will open the Ubuntu boot menu.



When booting from the USB drive, select **Try Ubuntu** instead of installing. This boots directly into a live environment without modifying your disk.



For detailed step-by-step instructions, refer to the Ubuntu Live USB tutorial.

After booting, you should be at the Ubuntu desktop. You may launch the terminal and continue as shown below.

Note

The screen resolution may be lower than you expect. This is a known issue, and it does not affect this procedure.

To access your files, you may run the following commands to mount and access your root file system on your NVMe drive:

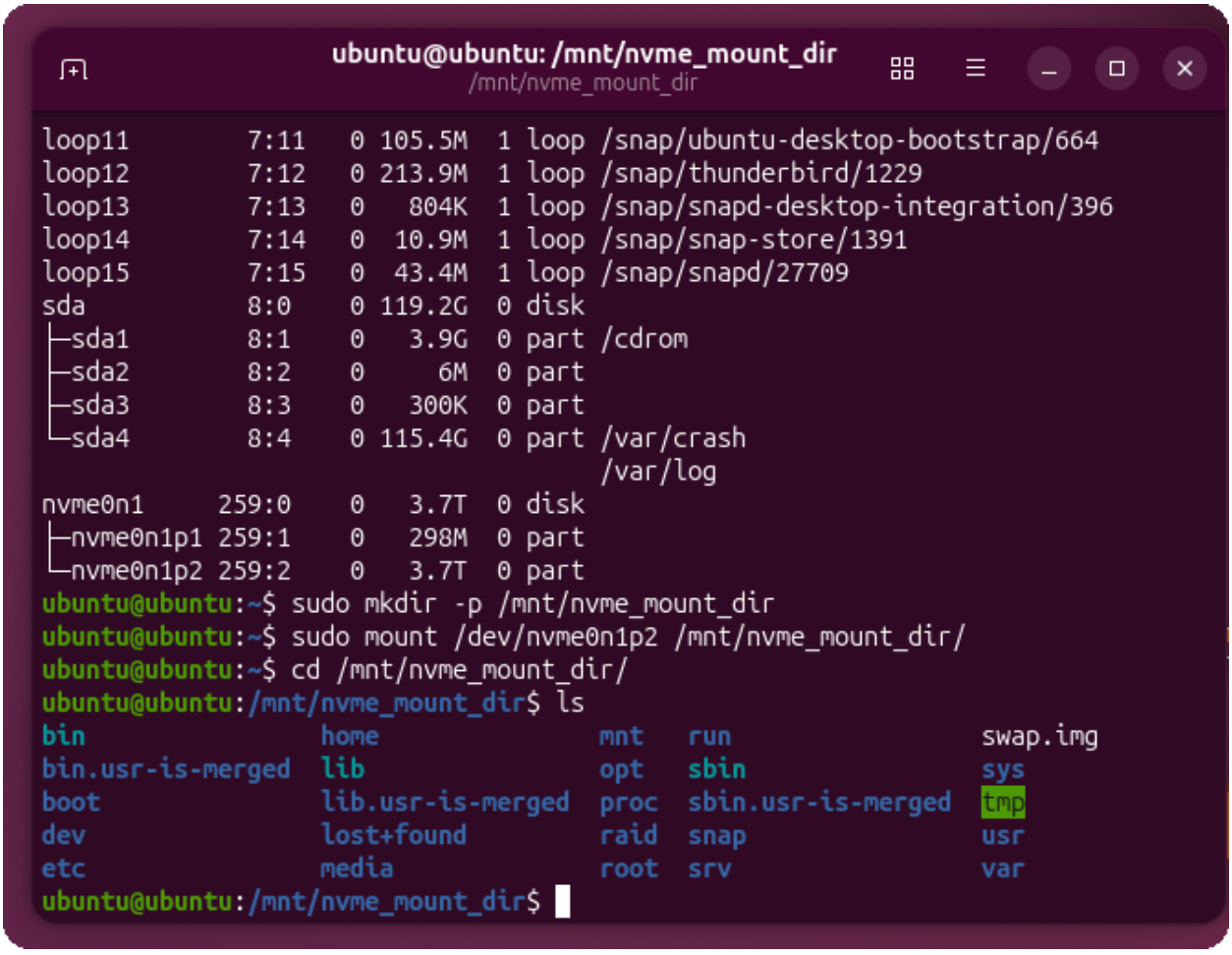
```
# Find your NVMe drive name.
lsblk

# Create a directory to access your nvme. The example uses "nvme_mount_dir"
↪under /mnt/
sudo mkdir -p /mnt/nvme_mount_dir

# Mount your nvme drive to the directory you just created
sudo mount /dev/nvme0n1p2 /mnt/nvme_mount_dir
```

You may now access your NVMe drive at `/mnt/nvme_mount_dir/` and back up your files by copy-

ing them to a separate external drive or a network location. User files are typically at `/mnt/nvme_mount_dir/home/`. Verify the backup before proceeding.



2.11.2.3 Common Scenarios for the Live USB

The examples below assume:

- Root partition: `/dev/nvme0n1p2`
- EFI partition: `/dev/nvme0n1p1`
- Mount location: `/mnt`

2.11.2.3.1 Scenario 1: Complete a Package Upgrade

This will help if you ran `apt upgrade` instead of `apt dist-upgrade` or need to update a faulty kernel. Do the following to update your configuration and complete the upgrade:

1. Mount your device:

```
sudo mount /dev/nvme0n1p2 /mnt # root partition
sudo mount /dev/nvme0n1p1 /mnt/boot/efi # EFI partition (mount a separate
↪/boot first if you have one)
for d in dev dev/pts proc sys run; do sudo mount --bind /$d /mnt/$d; done
```

(continues on next page)

(continued from previous page)

```
sudo mount --bind /sys/firmware/efi/efivars /mnt/sys/firmware/efi/efivars
↪2>/dev/null
sudo chroot /mnt /bin/bash
```

2. Update your configuration and complete the upgrade:

```
dpkg --configure -a
# Fix apt upgrade error
apt update
apt -f install
apt dist-upgrade
```

3. Unmount your device:

```
exit
sync
sudo umount -R /mnt
```

4. Reboot your device:

```
sudo reboot
```

2.11.2.3.2 Scenario 2: Rebuild initramfs

If you have an incomplete or missing initramfs, you can run the following to complete a half-finished upgrade:

1. Mount your device:

```
sudo mount /dev/nvme0n1p2 /mnt # root partition
sudo mount /dev/nvme0n1p1 /mnt/boot/efi # EFI partition (mount a separate
↪/boot first if you have one)
for d in dev dev/pts proc sys run; do sudo mount --bind /$d /mnt/$d; done
sudo mount --bind /sys/firmware/efi/efivars /mnt/sys/firmware/efi/efivars
↪2>/dev/null
sudo chroot /mnt /bin/bash
```

2. The following will help you complete a half-finished upgrade:

```
dpkg --configure -a
# Run the following command if you have an interrupted initramfs upgrade
update-initramfs -u -k all
# If you are missing your initramfs entirely, you need to regenerate it.
↪Run 'update-initramfs -c -k all' instead
update-grub
```

3. Unmount your device:

```
exit
sync
sudo umount -R /mnt
```

4. Reboot your device:

```
sudo reboot
```

2.11.2.3.3 Scenario 3: Install signed NVIDIA driver packages

Use this procedure if your NVIDIA GPU driver is missing or Secure Boot prevents its kernel modules from loading. The following steps remove leftover modules and install compatible, signed NVIDIA driver packages.

1. Mount your device:

```
sudo mount /dev/nvme0n1p2 /mnt # root partition
sudo mount /dev/nvme0n1p1 /mnt/boot/efi # EFI partition (mount a separate
↪/boot first if you have one)
for d in dev dev/pts proc sys run; do sudo mount --bind /$d /mnt/$d; done
sudo mount --bind /sys/firmware/efi/efivars /mnt/sys/firmware/efi/efivars
↪2>/dev/null
sudo chroot /mnt /bin/bash
```

2. Set KVER to the installed kernel version you want to repair and boot:

```
KVER=<kernel version>
```

3. Remove leftover modules from a previous installation.

   If you previously installed the NVIDIA driver with a `.run` installer, run:

```
nvidia-uninstall
```

   If you installed the driver only through APT, skip that command.

   If a previous manual installation left NVIDIA modules behind, remove files that are not owned by a Debian package. Confirm that KVER is set to the installed kernel you intend to repair before continuing:

```
find "/lib/modules/${KVER:?Set KVER first}" \
-name 'nvidia*.ko*' \
-exec sh -c 'dpkg -S "$1" >/dev/null 2>&1 || rm -v "$1"' _ {} \;
depmod -a "$KVER"
```

4. Find and install the signed packages:

```
apt update
apt-cache search linux-modules-nvidia | grep -F -- "$KVER"
apt-cache search nvidia-driver | grep -i open
# Set the corresponding modules and drivers to be the ones for your
↪selected kernel version
apt install <signed linux-modules-nvidia package> <nvidia-driver package>
```

> **Warning**
>
> Do not install `nvidia-dkms-*` modules as they are unsigned. The DGX Spark has Secure Boot enabled, so these modules will fail to load unless they are signed with a trusted key.

5. Rebuild and verify:

```
update-initramfs -u -k $KVER
modinfo -k $KVER -F signer nvidia # must be non-empty
```

6. Unmount your device:

```
exit
sync
sudo umount -R /mnt
```

7. Reboot your device:

```
sudo reboot
```

2.11.2.3.4 Scenario 4: Repair a Corrupted Filesystem

Run these steps from a healthy operating system, either by booting the affected computer from a live USB or by connecting the affected drive to another computer. This guide assumes you are using an Ubuntu live USB and have selected **Try Ubuntu**.

1. Identify the root partition and its filesystem type, then unmount it:

```
lsblk -f
sudo umount /dev/nvme0n1p2
```

If it reports “not mounted”, continue. If it reports “target is busy”, stop and close anything using that partition. Do not proceed until this partition is unmounted.

```
sudo e2fsck -f /dev/nvme0n1p2
```

Review and answer the repair prompts. **Never run this repair on a mounted filesystem.** The `e2fsck` command applies to ext2, ext3, and ext4 filesystems. For other filesystem types, use the appropriate repair tool as described in the Ubuntu e2fsck manual.

2. Once the check completes without unresolved errors, reboot and remove the USB:

```
sudo reboot
```

2.12. System Recovery

This section provides information about system recovery procedures for your DGX Spark.

Note

Founders Edition Only: This recovery information applies only to the DGX Spark Founders Edition. The Founders Edition OS is not distributed as an ISO; recovery media is provided as a tar.gz archive. OEM variant images are only available from the respective OEMs. Devices from other manufacturers might have different recovery procedures and recovery media.

2.12.1. System Recovery Overview

The system recovery process for the DGX Spark restores the operating system and firmware to their original factory state. This process erases the internal SSD. To repair the installed system or back up files without erasing the SSD, refer to *System Repair*.

2.12.1.1 Recovery Requirements

Before beginning the recovery process, ensure you have the following:

- A USB flash drive with 16GB or larger capacity
- A keyboard and display connected to your DGX Spark
- Access to download the recovery media from developer.nvidia.com

2.12.1.2 Recovery Process Steps

The recovery process involves downloading recovery media, creating a bootable USB drive, and following a series of UEFI configuration steps. Follow these steps carefully to ensure successful system recovery.

1. **Download Recovery Media**:
   - Download the recovery media archive file (tar.gz format) from DGX Spark System Recovery Image
   - Save the file to your local system
2. **Create Recovery USB Drive**:
   - Download and unzip the recovery media archive file
   - Insert the USB drive into your system
   - Within the extracted files, use one of the following commands to create the recovery drive:

> **Note**
>
> **Creating the recovery drive requires administrator privileges.** On Windows systems, this is typically done by running the command as an administrator (i.e. from an elevated PowerShell or Command Prompt). If supported by the host system, you will be prompted by the system to run the command with elevated privileges.

Windows:

```
CreateUSBKey.cmd
```

Linux:

```
CreateUSBKey.sh
```

MacOS:

```
CreateUSBKeyMacOS.sh
```

> **Warning**
>
> This process will erase all data on your USB drive. Ensure you have backed up any important data before proceeding.

3. **Boot from Recovery USB Drive**:
   - Disconnect any external storage devices from the DGX Spark
   - Connect the USB drive to one of the USB ports on your DGX Spark
   - If your DGX Spark is powered off, boot the device into UEFI settings by holding down the `Esc` or `Del` key immediately after powering on the device

   > **Important**
   >
   > Make sure to use a keyboard plugged directly into a USB port on the DGX Spark device. Some Bluetooth keyboards, even those with USB connections, may not work during this process. If your keyboard isn't recognized, use a standard USB keyboard.

4. **Restore UEFI Defaults**:
   - Tap the `Right Arrow` key to select the "Save & Exit" page within the UEFI settings
   - Tap `Down Arrow` to select "Restore Defaults" and select "Yes" in response to "Load Optimized Defaults"
   - Select "Save Changes and Reset" - the device will reboot
   - While the device is rebooting, hold down the `Esc` or `Del` key to re-enter UEFI settings a second time
5. **Boot from Recovery Media**:
   - Tap the `Right Arrow` key to select the "Save & Exit" page in UEFI settings
   - Tap `Down Arrow` to move to the "Boot Override" section
   - Select the USB drive and tap `Enter`
   - The device will reboot using the USB drive - follow the on-screen steps to update your firmware and re-install the OS on the device's SSD

   > **Note**
   >
   > If the USB drive appears under **Boot Override** but the system reboots into the operating system:
   >
   > 1. Go to **Security** → **Secure Boot** and set **Secure Boot** to **Disabled**.
   > 2. Save changes and reset.
   > 3. Re-enter UEFI and use **Boot Override** to boot from the USB drive.
   >
   > After recovery completes, Secure Boot is re-enabled automatically.

6. **Restore System from Recovery Environment**

- On the welcome screen, press `Enter` to continue. To cancel the process, press Esc.



- On the warning screen, select [EXIT] to restart without proceeding, or select [START RECOVERY] to begin reflashing the SSD. Be aware that this will completely erase the internal SSD on the DGX Spark.



- Monitor the Recovery Progress screen for detailed output from the recovery process and the progress of the SSD reflash.



- When the process is complete, review the progress screen and follow the prompt to continue. Scroll or page through the recovery output if needed, for example during customer support.



- On the final screen, confirm that the DGX Spark has been reset to factory state and press `Enter` to restart the device.



2.13. Get the Right Support for Your DGX Spark

Choose the options that match your needs.

Important

Use of the NVIDIA CUDA Toolkit preinstalled on DGX Spark is subject to the NVIDIA CUDA Toolkit End User License Agreement.

2.13.1. Passthrough Virtualization Support

GPU passthrough for virtualization environments is not supported.

2.13.2. NVIDIA DGX Spark Hardware Support

- Open a new ticket. Log In
- Chat online with our support agents. Chat Now
- Log in to view your existing tickets. View Tickets
- Join the NVIDIA DGX Spark Forums to ask questions, share experiences, and get help from other users and NVIDIA engineers.

2.13.3. NVIDIA AI Enterprise—DGX Spark Support

Enterprise technical support for the NVIDIA AI Enterprise—DGX Spark software stack on DGX Spark requires an entitlement for **NVIDIA AI Enterprise—DGX Spark**. Other NVIDIA AI Enterprise entitlements do not automatically include this product or its support. Confirm that your NVIDIA Entitlement Certificate lists **NVIDIA AI Enterprise—DGX Spark** before opening enterprise cases for DGX Spark software.

- Open a new ticket. Log In
- Chat online with our support agents. Chat Now
- Log in to view your existing tickets. View Tickets
- For registration, downloads, and portal access, see the NVIDIA AI Enterprise—DGX Spark Quick Start Guide: *NVIDIA AI Enterprise—DGX Spark Quick Start Guide*.

2.13.4. Enterprise Manageability

For fleet lifecycle integration, platform integration patterns, and custom installation with cloud-init, refer to *Enterprise Manageability*.

2.13.5. Security and Vulnerability Response

For NVIDIA's approach to receiving, assessing, and disclosing product security issues, see the NVIDIA Product Security Incident Response Team (PSIRT) policies.

2.13.6. Field Diagnostic Software

NVIDIA Field Diagnostic is a software program used to test the DGX Spark system and detect hardware failures, and is intended for health checks of your DGX Spark setup and a pre-check for RMA qualification of the overall system.

For complete instructions, refer to the `Field Diagnostics User Guide`.

2.13.6.1 Removing a Previous Version

Remove the previous version of the field diagnostic software before installing a new one with the following commands:

```
sudo dpkg -P dgx-spark-fieldiag
sudo rm -rf /opt/nvidia/dgx-spark-fieldiag
sudo apt autoremove dgx-spark-fieldiag
```

2.13.6.2 Package Dependencies

When you install the `dgx-spark-fieldiag` package, `stress-ng`, `fio`, and `memtester` are installed automatically as package dependencies. These tools support the core CPU, SSD, and memory stress tests. This automatic installation does not include the ConnectX-7 (CX7Stress) test tools.

2.13.6.3 ConnectX-7 Test Prerequisites

The CX7Stress test requires additional tools that are not installed with the `dgx-spark-fieldiag` package. Before you run field diagnostics, install DOCA OFED and the NVIDIA Firmware Tools (MFT) kernel modules on your system. For installation and removal instructions, refer to the Field Diagnostics User Guide.

2.13.6.4 Installing the Field Diagnostic Software

The Field Diagnostic software package is named `dgx-spark-fieldiag_<version>-1_arm64.deb`. To install the package, follow these steps:

1. Install the package:

```
sudo apt-get update
sudo apt-get install dgx-spark-fieldiag
```

2. Verify the installation:

```
dpkg -l | grep dgx-spark-fieldiag
```

2.13.6.5 Verifying Tool Installation

To verify that the core diagnostic tools are installed, run the following commands. Each command should display the path to the tool binary.

```
which fio
which memtester
which stress-ng
```

To verify that the ConnectX-7 (CX7Stress) tools are installed, run the following commands:

```
which ofed_info
which opensm
which ibstat
which mlxlink
```

If a command returns no output, install the missing packages before you run field diagnostics. Refer to the Field Diagnostics User Guide for ConnectX-7 prerequisite installation and removal instructions.

2.13.6.6 Running Field Diagnostics

After installing the package, the field diagnostic software is located at `/opt/nvidia/dgx-spark-fieldiag`.

2.13.6.6.1 Before Running

Disable Secure Boot before running field diagnostics:

1. Check the current Secure Boot state:

```
sudo mokutil --sb-state
```

2. Reboot and enter UEFI setup (press **Delete** during boot, or run `sudo systemctl reboot --firmware-setup`).

3. Go to **Security** → **Secure Boot** → **Disable Secure Boot**.
4. Save the changes and reboot the system.

2.13.6.6.2 Running the Diagnostic

To execute field diagnostics, use root access:

1. Run:

```
sudo init 3
```

2. When the system switches to TTY console mode, log in at the TTY console.
3. Run:

```
cd /opt/nvidia/dgx-spark-fieldiag
sudo ./partnerdiag --field
```

The diagnostic takes approximately 30 minutes. A PASS/FAIL banner appears when complete. You can also run the diagnostic over SSH using the same commands.

> **Note**
>
> If you interrupt the diagnostic (for example, with Ctrl+C), power cycle the system before running the tests again.

2.13.6.6.3 After Running

Re-enable Secure Boot after the diagnostic completes:

1. Run `sudo systemctl reboot --firmware-setup`.
2. Go to **Security** → **Secure Boot** → **Enable Secure Boot**.
3. Save the changes and reboot the system.

For detailed instructions, including the Spec JSON file and log retrieval, refer to the Field Diagnostics User Guide.

2.14. Regulatory Compliance Information

2.14.1. Safety Information

NVIDIA products are designed to operate safely when installed and used according to the product instructions and general safety practices. The guidelines included in this document explain the potential risks associated with equipment operation and provide important safety practices designed to minimize these risks. By carefully following the information contained in this document, you can protect yourself from hazards and create a safer environment.

This product is designed and tested to meet IEC 62368-1 (Standard for Safety Audio/video, information and communication technology equipment). This also covers the national implementation of IEC 62368-1 based safety standards around the world, for example UL 62368-1. These standards reduce the risk of injury from the following hazards:

- Electric shock: Hazardous voltage levels contained in parts of the product
- Fire: Overload, temperature, material flammability
- Mechanical: Sharp edges, moving parts, instability
- Energy: Circuits with high energy levels (240-volt amperes) or potential as burn hazards
- Heat: Accessible parts of the product at high temperatures
- Chemical: Chemical fumes and vapors
- Radiation: Noise, ionizing, laser, ultrasonic waves

Retain and follow all product safety and operating instructions. Always refer to the documentation supplied with your equipment. Observe all warnings on the product and in the operating instructions.

Note

Read all the safety information below and operating instructions before using DGX Spark to avoid injury.

Warning

FAILURE TO FOLLOW THESE SAFETY INSTRUCTIONS COULD RESULT IN FIRE, ELECTRIC SHOCK OR OTHER INJURY OR DAMAGE. ELECTRICAL EQUIPMENT CAN BE HAZARDOUS IF MISUSED. OPERATION OF THIS PRODUCT, OR SIMILAR PRODUCTS, MUST ALWAYS BE SUPERVISED BY AN ADULT. DO NOT ALLOW CHILDREN ACCESS TO THIS ELECTRICAL PRODUCT AND DO NOT PERMIT THEM TO HANDLE ANY CABLES.

To reduce the risk of bodily injury, electric shock, fire, and damage to the equipment, observe the safety labels included on the equipment.

2.14.1.1 Anti-Static Precautions

- Avoid working in areas where static electricity can build up, such as carpeted areas.
- To minimize the possibility of a static charge harming the hardware, always use a properly grounded anti-static wrist strap when working with DGX Spark.
- In addition, always discharge any residual static electricity that may be present by touching a non-conducting edge of the kit first. Never touch components, pins, or connectors without the wrist strap attached and grounded.

2.14.1.2 Symbols on Equipment



This is a universal recycling symbol indicating that the electronic device cannot be disposed of in the trash and must be recycled according to the regulations and/or ordinances of the local community.



Caution: this symbol indicates that the marked item can be hot and should not be touched without taking care.

2.14.1.3 General Precautions

To reduce the risk of electric shock, personal injury, or damage to the equipment, observe the following precautions:

- Follow all cautions and instructions marked on the equipment.
- Do not eat, drink, or smoke while working with DGX Spark.
- Do not bend, drop, crush, puncture, incinerate, or push objects into DGX Spark.
- Do not connect or disconnect any cables or perform maintenance or reconfiguration of this product during an electrical storm or when the device is powered on.
- Do not use DGX Spark in rain or near a sink or other wet locations. Take care not to spill any food or liquid into the device. If DGX Spark gets wet, unplug all cables and turn off before cleaning, and allow it to dry thoroughly before turning it on again.
- Do not expose DGX Spark to any gas which is not normally contained in the atmosphere; it may cause defects.
- Never turn on any equipment when there is evidence of fire, water, or structural damage.
- Place the product away from radiators, heat registers, stoves, amplifiers, or other appliances that produce heat.
- Never force a connector into a port. Check for obstructions on the port. If the connector and port do not join with reasonable ease, they probably do not match. Make sure that the connector matches the port and that you have positioned the connector correctly in relation to the port.
- Always use an NVIDIA approved power adapter and refer to the manufacturer's instructions or equipment manual for proper instructions. Do not connect or disconnect the AC power adapter with wet hands. The power adapter is intended for indoor use only.
- DGX Spark shall be connected to a grounding terminal. Plug the system power cord into an AC outlet that provides a ground connection. Use only the provided AC adapter and power cord. Use of any other AC adapter or power cord may present risk of fire or electric shock.
- Use certified Optical Fiber Transceiver Class I Laser Product.
- Operate the AC adapter in a ventilated area.
- When you are using DGX Spark, it is normal for the system and/or AC power adapter to get warm.
- Do not use conductive tools that could bridge live parts.
- Do not make mechanical or electrical modifications to the equipment.
- Operate and store DGX Spark in a place where the temperature is always between 5 and 30°C (41 to 86°F).
- Use this product only with approved equipment and/or accessories.
- If the product sustains damage requiring service, disconnect the product from the AC electrical outlet and refer servicing to an NVIDIA authorized service provider.
- This equipment is not suitable for use in locations where children are likely to be present.

For service personnel only:

CAUTION - Risk of fire or explosion if the battery is replaced by an incorrect type. Dispose of used batteries according to the instructions.

2.14.1.4 Power Rating

48V 5A

2.14.1.5 USB-C Ports Rating

Each USB-C port can be loaded 5V/3A respectively. The total output power of USB-C ports is limited to 30W.

2.14.2. Compliance

DGX Spark (Model: P4242) is compliant with the regulations listed in this section.

2.14.2.1 United States

2.14.2.1.1 Federal Communications Commission (FCC)



Contains FCC ID: RAS-MT7925B14L

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including any interference that may cause undesired operation of the device.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

FCC Warning: The FCC requires that you be notified that any changes or modifications to this device not expressly approved by the manufacturer could void the user's authority to operate the equipment. Only those antennas with same type and lesser/equal gain filed under this FCC ID number can be used with this device.

- Transmitters in the 5.925-7.125 GHz band are prohibited from operating to control or communicate with unmanned aircraft systems.

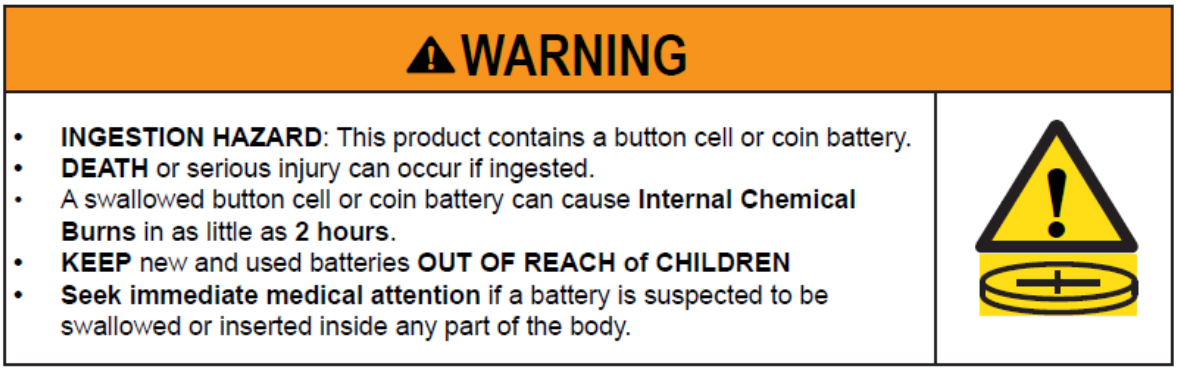
RF Radiation Exposure Statement:

DGX Spark has been tested and complies with FCC RF radiation exposure limits set forth for an uncontrolled environment when used with the NVIDIA accessories supplied or designated for this product. During testing, DGX Spark was placed in positions that simulate use against the body. The use of any other accessories may not ensure compliance with FCC RF exposure guidelines.

This equipment must be installed and operated in accordance with provided instructions and the antenna(s) used for this transmitter must be installed to provide a separation distance of at least 20 cm from all persons and must not be co-located or operated in conjunction with any other antenna or transmitter.

2.14.2.1.2 Underwriters Laboratories (UL)





The product contains a non-replaceable coin battery, CR2032 with a nominal voltage of 3 Vdc. This battery is a non-rechargeable battery and must not be recharged.

Remove and immediately recycle or dispose of used batteries according to local regulations and keep them away from children. Do NOT dispose of batteries in household trash or incinerate them. Even used batteries may cause severe injury or death. Call a local poison control center for treatment information if needed.

Do not force discharge, recharge, disassemble, heat above the operating temperature, or incinerate batteries. Doing so may result in injury due to venting, leakage, or explosion, which can cause chemical burns.

2.14.2.2 Canada

2.14.2.2.1 Innovation, Science and Economic Development Canada (ISED)

Contains IC: 7542A-MT7925B14L

CAN ICES (B)/NMB (B)

This device complies with Industry Canada's license-exempt RSSs. Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Caution:

Operation in the band 5150–5250 MHz is only for indoor use to reduce the potential for harmful interference to co-channel mobile satellite systems.

Be advised that high-power radars are allocated as primary users of the bands 5250-5350 MHz and 5650-5850 MHz and that these radars could cause interference and/or damage to LE-LAN devices.

Operation in the band 5925-7125 MHz, the following restrictions must be met:

- Devices shall not be used for control of or communications with unmanned aircraft systems.
- Devices shall not be used on oil platforms.
- Devices shall not be used on aircraft, except for the low-power indoor access points, indoor subordinate devices, low-power client devices, and very low-power devices operating in the 5925-6425 MHz band, that may be used on large aircraft as defined in the Canadian Aviation Regulations, while flying above 3,048 metres (10,000 feet).
- Devices shall not be used on automobiles.
- Devices shall not be used on trains.
- Devices shall not be used on maritime vessels.

Attention:

Les dispositifs fonctionnant dans la bande de 5 150 à 5 250 MHz sont réservés uniquement pour une utilisation à l'intérieur afin de réduire les risques de brouillage préjudiciable aux systèmes de satellites mobiles utilisant les mêmes canaux.

Être informé des radars à haute puissance sont alloués principaux utilisateurs de bandes 5250 à 5350 MHz et 5850 MHz, à 5650 et que ces radars pourraient provoquer des interférences et / ou endommager les appareils RL-EL.

Pour fonctionner dans la bande 5925-7125 MHz, les restrictions suivantes doivent être respectées:

- Les dispositifs ne doivent pas être utilisés pour commander des systèmes d'aéronef sans pilote ni pour communiquer avec de tels systèmes.
- Les dispositifs ne doivent pas être utilisés sur les plateformes de forage pétrolier.
- Les dispositifs ne doivent pas être utilisés dans les aéronefs, à l'exception des points d'accès intérieurs de faible puissance, des dispositifs subordonnés intérieurs, des dispositifs clients de faible puissance et des dispositifs de très faible puissance fonctionnant dans la bande de 5925 à 6425 MHz, qui peuvent être utilisés dans les gros aéronefs, tel qu'il est défini dans le Règlement de l'aviation canadien, lorsqu'ils volent à une altitude supérieure à 3 048 mètres (10 000 pieds).

- Les dispositifs ne doivent pas être utilisés dans les automobiles.
- Les dispositifs ne doivent pas être utilisés dans les trains.
- Les dispositifs ne doivent pas être utilisés sur les navires maritimes.

RF Radiation Exposure Statement:

DGX Spark has been tested and complies with IC RSS 102 RF radiation exposure limits set forth for an uncontrolled environment when used with the NVIDIA accessories supplied or designated for this product. During testing, the DGX Spark was placed in positions that simulate use against the body. The use of any other accessories may not ensure compliance with IC RSS 102 RF exposure guidelines.

This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter. This equipment should be installed and operated with a minimum distance of 20 centimeters between the radiator and your body.

Déclaration d'exposition aux radiations:

NVIDIA DGX Spark a ete testee conformemment aux normes d'exposition d'emission RF de la IC RSS 102 pour un environement non controle lors d'utilisation avec les accessoires fournis or recommendes par NVIDIA. Pendant les tests, le DGX Spark a ete positionne afin de simule la presence du corps humain. L'utilisation d'accessoires autres que ceux recommendes par NVIDIA ne guarantis pas la compatibilite avec les normes d'emission RF de la IC RSS 102.

Cet émetteur ne doit pas être Co-placé ou ne fonctionnant en même temps qu'aucune autre antenne ou émetteur. Cet équipement devrait être installé et actionné avec une distance minimum de 20 centimètres entre le radiateur et votre corps.

2.14.2.3 European Union

2.14.2.3.1 European Conformity; Conformité Européenne (CE)



	BE	BG	CZ	DK	DE	EE	IE	EL	ES	FR	HR
	IT	CY	LV	LT	LU	HU	MT	NL	AT	PL	PT
	RO	SI	SK	FI	SE	UK(NI)	IS	LI	NO	CH	TR

This device complies with the following Directives:

- Radio Equipment Directive 2014/53/EU
- RoHS 2011/65/EU as amended by 2015/863/EU
- Energy-related Products Directive (ErP) 2009/125/EC

The full text of Declaration of Conformity is available at the following internet address: http://www.nvidia.com/support

A copy of the Declaration of Conformity to the essential requirements may also be obtained directly from NVIDIA GmbH (Bavaria Towers – Blue Tower, Einsteinstrasse 172, D-81677 Munich Germany).

This device operates in the following frequency bands and maximum transmitted power:

Frequency Band	Max. EIRP (dBm)	Max. EIRP (mW)
2400.0 - 2483.5 MHz	20.0	100
5150.0 - 5350.0 MHz	23.0	200
5470.0 – 5725.0 MHz	23.0	200
5725.0 – 5875.0 MHz	14.0	25
5925.0 – 6425.0 MHz (LPI mode)	23.0	200
5925.0 – 6425.0 MHz (VLP mode)	14.0	25

Warning:

- Operation in the 5150 – 5350 MHz and 5925 – 6425 MHz (LPI mode) frequency bands are restricted to Indoor Use Only.
- This equipment should be installed and operated with a minimum distance of 20 centimeters between the radiator and your body.
- Any changes or modifications to this device not expressly approved by the manufacturer could void the user's authority to operate the equipment.

Hardware version: 670-54242-0000

Firmware version: 5.36

Exposed services via network interfaces:

Port	State	Service
22/tcp	Open	OpenSSH
80/tcp	Open	HTTP

2.14.2.3.2 Commission Regulation (EU) No 617/2013 Technical information

DGX Spark, Model P4242, is classified as a small-scale server, and manufactured by NVIDIA Corporation (Santa Clara, CA, 95051, USA) in 2025.

This device is in compliance with (EU) No 617/2013.

Maximum Power (W)	233.2
Idle State Power (W)	38.0
Off Mode Power (W)	4.1
Test methodology	IEC 62623 / EN 62623
Test Voltage and Frequency	230V / 50Hz
Total Harmonic Distortion	< 2%
Noise Levels (per ECMA-74)	Refer to DGX Spark support page
External power supply efficiency (per EN50563)	Level VI

2.14.2.4 Great Britain (England, Wales and Scotland)

2.14.2.4.1 UK Conformity Assessed



2.14.2.4.2 Restrictions or Requirements for use in the UK

This device complies with the following Regulations:

- SI 2017/1206 Radio Equipment Regulations 2017
- SI 2012/3032: The Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment (As Amended)
- SI 2010/2617: The Ecodesign for Energy-Related Products Regulations 2010

The full text of Declaration of Conformity is available at the following internet address: http://www.nvidia.com/support

A copy of the Declaration of Conformity to the essential requirements may be obtained directly from NVIDIA Ltd. (100 Brook Drive, 3rd Floor Green Park, Reading RG2 6UJ, United Kingdom).

This device operates in the following frequency bands and maximum transmitted power:

Frequency Band	Max. EIRP (dBm)	Max. EIRP (mW)
2400.0 - 2483.5 MHz	20.0	100
5150.0 - 5350.0 MHz	23.0	200
5470.0 – 5725.0 MHz	23.0	200
5725.0 – 5875.0 MHz	14.0	25
5925.0 – 6425.0 MHz (LPI mode)	23.0	200
5925.0 – 6425.0 MHz (VLP mode)	14.0	25

Warning:

- Operation of this device in the 5150 - 5350 MHz and 5925 – 6425 MHz (LPI mode) frequency bands are restricted to indoor use.
- This equipment must be installed and operated with a minimum distance of 20 cm between the radiator and your body.
- Any modifications to this device, not expressly approved by the manufacturer, voids the user authority to operate the equipment.

2.14.2.5 Australia and New Zealand

2.14.2.5.1 Australian Communications and Media Authority

This product meets the applicable EMC requirements for Class B, I.T.E equipment and applicable radio equipment requirements.

This product meets the applicable requirements for telecommunications equipment and can legally be connected to a telecommunications network.

2.14.2.6 Japan

2.14.2.6.1 Voluntary Control Council for Interference (VCCI)

この装置は、クラスB機器です。この装置は、住宅環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。

取扱説明書に従って正しい取り扱いをして下さい。　VCCI－B

2.14.2.6.2 Radio/Telecommunications Certification

R 020-240073

T D240014020

5.2GHz, 5.3GHz □□ 6GHz □ (LPI): □□□□□□

□□

2.14.2.6.3 Japan RoHS Material Content Declaration

日本工業規格 JIS C 0950:2008 により、2006 年 7 月 1 日以降に販売される特定分野の電気および電子機器について、製造者による含有物質の表示が義務付けられます。

機器名称：小規模サーバー

主な分類	特定化学物質記号					
	Pb	Hg	Cd	Cr(VI)	PBB	PBDE
PCB ボード	0	0	0	0	0	0
機械部品及びファン	除外項目	0	0	0	0	0
コネクター	除外項目	0	0	0	0	0
パッシブ電子部品	除外項目	0	0	0	0	0
プロセッサー	0	0	0	0	0	0
アクティブ電子部品	除外項目	0	0	0	0	0
メモリ	0	0	0	0	0	0
ケーブル	除外項目	0	0	0	0	0
はんだ付け材料	0	0	0	0	0	0
フラックス、クリームはんだ、ラベル、その他消耗品	0	0	0	0	0	0

注:

1.「0」は、特定化学物質の含有率が日本工業規格 JIS C 0950:2008 に記載されている含有率基準値より低いことを示します。

2.「除外項目」は、特定化学物質が含有マークの除外項目に該当するため、特定化学物質について、日本工業規格 JIS C 0950:2008 に基づく含有マークの表示が不要であることを示します。

3.「0.1wt%超」または「0.01wt%超」は、特定化学物質の含有率が日本工業規格 JIS　C　0950:2008 に記載されている含有率基準値を超えていることを示します。

A Japanese regulatory requirement, defined by specification JIS C 0950: 2008, mandates that manufacturers provide Material Content Declarations for certain categories of electronic products offered for sale after July 1, 2006.

Product Name: Small-Scale Server

Major Classification	Symbols of Specified Chemical Substance					
	Pb	Hg	Cd	Cr(VI)	PBB	PBDE
PCB	0	0	0	0	0	0
Mechanical parts and Fan	Exempt	0	0	0	0	0
Connectors	Exempt	0	0	0	0	0
Passive components	Exempt	0	0	0	0	0
Processor	0	0	0	0	0	0
Active components	Exempt	0	0	0	0	0
Memory	0	0	0	0	0	0
Cables	Exempt	0	0	0	0	0
Soldering matel	0	0	0	0	0	0
Flux, solder paste, label and other consumable materials	0	0	0	0	0	0

Notes:

1. "0" indicates that the level of the specified chemical substance is less than the threshold level specified in the standard, JIS C 0950: 2008.

2. "Exempt" indicates that the specified chemical substance is exempt from marking and it is not required to display the marking for that specified chemical substance per the standard, JIS C 0950: 2008.

3. "Exceeding 0.1wt%" or "Exceeding 0.01wt%" is entered in the table if the level of the specified chemical substance exceeds the threshold level specified in the standard, JIS C 0950: 2008.

2.14.2.7 China

2.14.2.7.1 China Compulsory Certificate



2.14.2.7.2 State Radio Regulation of China (SRRC)

CMIIT ID: 25J99Y4B4242

2.14.2.7.3 China RoHS Material Content Declaration





产品中有害物质的名称及含量
根据中国《电器电子产品有害物质限制使用管理办法》

部件名称	有害物质									
	铅 (Pb)	汞 (Hg)	镉 (Cd)	六价铬 (Cr(VI))	多溴联苯 (PBB)	多溴二苯醚 (PBDE)	邻苯二甲酸二正丁酯 (DBP)	邻苯二甲酸二异丁酯 (DIBP)	邻苯二甲酸丁基苄酯 (BBP)	邻苯二甲酸二（2-乙基）己酯 (DEHP)
印刷电路板	0	0	0	0	0	0	0	0	0	0
机械部件以及风扇	X	0	0	0	0	0	0	0	0	0
连接器	X	0	0	0	0	0	0	0	0	0
被动电子零件	X	0	0	0	0	0	0	0	0	0
中央处理器	0	0	0	0	0	0	0	0	0	0
主动电子零件	X	0	0	0	0	0	0	0	0	0
内存	0	0	0	0	0	0	0	0	0	0
线材	X	0	0	0	0	0	0	0	0	0
焊接金属	0	0	0	0	0	0	0	0	0	0
助焊剂，锡膏，标签及其他耗材	0	0	0	0	0	0	0	0	0	0

本表格依据 SJ/T 11364-2024 的规定编制

O：表示该有害物质在该部件所有均质材料中的含量均在 GB/T 26572 标准规定的限量要求以下。
X：表示该有害物质至少在该部件的某一均质材料中的含量超出 GB/T 26572 标准规定的限量要求。

此表中所有名称中含 “X” 的部件均符合欧盟 RoHS 立法。

注：环保使用期限的参考标识取决于产品正常工作的温度和湿度等条件

2.14.2.8 South Korea

2.14.2.8.1 Korea Certification (KC)



□□□: P4242
R-R-NVA-P4242
R-C-MD6-MT7925B14L

B 급 기기 (가정용 방송 통신기자재)	이 기기는 가정용(B 급) 전자파적합기기로서 주로 가정에서 사용하는 것을 목적으로 하며, 모든 지역에서 사용할 수 있습니다.

2.14.2.8.2 Korea RoHS Material Content Declaration

<table>
<tr><td colspan="5">확인 및 평가 양식은 제품에 포함 된 유해 물질의 허용 기준의 준수에 관한</td></tr>
<tr><td rowspan="3">문
준비</td><td>상호 :</td><td>앤비디아홍콩홀딩즈리미티드(영업소)</td><td>법인등록번호</td><td>110181-
0036373</td></tr>
<tr><td>대표자성명</td><td>카렌테레사번즈</td><td>사업자등록번호:</td><td>120-84-
06711</td></tr>
<tr><td>주소</td><td colspan="3">서울특별시 강남구 영동대로 511, 2101 호 (삼성동, 코엑스무역타워)</td></tr>
<tr><td colspan="5">제품 내용</td></tr>
<tr><td>제품의 종류</td><td>컴퓨터 서버</td><td>제품명(규격)</td><td colspan="2">소규모 서버</td></tr>
<tr><td>세부모델명(번호):</td><td>P4242</td><td>제품출시일</td><td colspan="2">해당없음</td></tr>
<tr><td>제품의 중량</td><td>해당없음</td><td>제조, 수입업자</td><td colspan="2">앤비디아</td></tr>
<tr><td colspan="5">엔비디아의 그래픽 카드제품은 전기 전자제품 및 자동차의 자원순환에 관한 법률 시행령 제 11 조 제 1 항에 의거한 법 시행행규칙 제 3 조에에따른 유해물질함유 기준을 확인 및 평가한 결과, 이를 준수하였음을 공표합니다.</td></tr>
<tr><td colspan="5">구비서류 : 없음
작성방법

① 제품의 종류는 "전기.전자제품 및 자동차의 자원순환에관한 법률 시행령" 제 8 조 제 1 항 및 제 2 항에 따른 품목별로 구분하여 기재합니다.

② 전기 전자 제품의 경우 모델명 (번호), 자동차의 경우, 제원관리번호를 기재합니다.

③ 해당제품의 제조업자 또는 수입업자를 기재합니다.</td></tr>
</table>

Confirmation and Evaluation Form Concerning the Adherence to Acceptable Standards of Hazardous Materials Contained in Products

Statement Prepared by	Company Name:	Nvidia HongKong Holding Ltd.Korea branch	Corporate Identification Number:	110181-0036373
	Name of Company Representative:	Karen Theresa Burns	Business Registration Number:	120-84-06711
	Address	2788 San Tomas Expressway, Santa Clara, CA 95051		

Product Information

Product Category:	Computer server	Name of Product:	Small-Scale Server
Detailed Product Model Name (Number):	P4242	Date of first market release:	N/A
Weight of Product:	N/A	Manufacturer and/or Importer:	NVIDIA Corporation

NVIDIA has evaluated and confirmed that the products comply with the standards for the content of hazardous substances in accordance with Article 3 of the 'Act on Resource Circulation of Electrical and Electronic Equipment and Vehicles' pursuant to the Enforcement Regulations of Article 11, Paragraph 1 in the statute.

Attachment: None

* Preparing the Form

① Please indicate the product category according to the categories listed in Article 8, Items 1and 2 of the ' Enforcement Ordinance of the Statute on the Recycling of Electrical, Electronic and Automobile Materials'

② For electrical and electronic products, please indicate the Model Name (and number). For automobiles, please indicate the Vehicle Identification Number.

③ Please indicate the name of manufacturer and/or importer of the product.

2.14.2.9 Taiwan

2.14.2.9.1 Bureau of Standards, Metrology & Inspection (BSMI)



This device complies with CNS 13438 (2006) Class B

□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□□80022300

□□□□□□□□□□□□ 8 □

2.14.2.9.2 Taiwan RoHS Material Content Declaration

Taiwan RoHS Material Content Declaration

限用物質含有情況標示聲明書 Declaration of the presence condition of the Restricted Sustances Marking						
設備名稱：輕量型伺服器 Equipment Name: Small-Scale Server						
單元 Parts	限用物質及其化學符號 Restricted substances and its chemical symbols					
	铅 (Pb)	汞 (Hg)	镉 (Cd)	六價铬 (Cr^{+6})	多溴聯苯 (PBB)	多溴二苯醚 (PBDE)
PCB 板 PCB	O	O	O	O	O	O
結構件以及風扇 Mechanical parts and Fan	-	O	O	O	O	O
連接器 Connectors	-	O	O	O	O	O
被動電子零件 Passive components	-	O	O	O	O	O
中央處理器 CPU	O	O	O	O	O	O
主動電子零件 Active components	-	O	O	O	O	O
內存 Memory	O	O	O	O	O	O
線材 Cables	-	O	O	O	O	O
焊接金屬 Soldering material	O	O	O	O	O	O
助焊劑，錫膏，標籤及耗材 Flux, Solder Paste, label and other consumable materials	O	O	O	O	O	O

備考 1：O：系指該限用物質未超出百分比含量基準值
Note 1： O：indicates that the percentage content of the restricted substance does not exceed the percentage of reference value of presence.
備考 2： -：系指該項限用物質為排外項目。
Note 2：-：indicates that the restricted substance corresponds to the exemption.

此表中所有名稱中含 "-" 的部件均符合歐盟 RoHS 立法。
All parts named in this table with an "-" are in compliance with the European Union's RoHS Legislation.

注：環保使用期限的參考標識取決與產品正常工作的温度和濕度等條件
Note: The referenced Environmental Protection Use Period Marking was determined according to normal operating use conditions of the product such as temperature and humidity.

2.14.2.9.3 National Communications Commission

國家通訊傳播委員會

CCAI24Y10040T0

□□□

□□ □□□□□□□□□□□□□□

□□□□

□□□

□□□□

□□□

□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□ 5.250~5.350GHz □□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

2.14.2.10 India

2.14.2.10.1 Bureau of Indian Standards (BIS)

Self Declaration - Conforming to IS 13252:2010, R-41315486

IS 13252 (Part 1)/
IEC 60950-1
R-41315486
www.bis.gov.in

Authenticity may be verified by visiting the Bureau of Indian Standards website at http://www.bis.gov.in

2.14.2.10.2 India RoHS compliance statement

This product, as well as its related consumables and spares, complies with the reduction in hazardous substances provisions of the “India E-waste (Management and Handling) Rule 2016”. It does not contain lead, mercury, hexavalent chromium, polybrominated biphenyls or polybrominated diphenyl ethers in concentrations exceeding 0.1 weight % and 0.01 weight % for cadmium, except for where allowed pursuant to the exemptions set in Schedule 2 of the Rule.

2.14.2.10.3 Wireless Planning & Coordination Wing (WPC)

Registration No.: ETA-SD-20250807136

2.14.2.11 Singapore

2.14.2.11.1 Infocomm Media Development Authority of Singapore (IMDA)

Complies with IMDA Standards DB100398

2.14.2.12 Malaysia

2.14.2.12.1 Malaysian Communications and Multimedia Commission (MCMC)



2.14.2.13 Bluetooth



The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NVIDIA Corporation is under license. Other trademarks and trade names are those of their respective owners.

2.15. IMPORTANT NOTICE - PLEASE READ AND AGREE BEFORE USING NVIDIA SOFTWARE OFFERINGS

This NVIDIA Software License Agreement is entered into between the entity you represent, or, if you are an Administrator deploying, configuring, installing, updating, enabling, or managing the Software Offerings for use by others, the entity on whose behalf you are taking such action, or, if you do not designate an entity, you individually (“Customer”), and NVIDIA Corporation (“NVIDIA”) and the Product Specific Terms (if applicable) govern the use of the Software Offerings. This NVIDIA Software License

Agreement consists of the terms and conditions below and all documents attached to or referenced in this NVIDIA Software License Agreement (together, the “Agreement”). Key terms are defined in Section 17. Customer and NVIDIA are each a “party” and collectively the “parties.”

By registering to use, using, or by an Administrator deploying, configuring, installing, updating, enabling, or managing the Software Offerings for use by others, Customer is affirming that it has read the Agreement and agrees to its terms. If Customer, or the individual acting on Customer’s behalf, does not have the required authority to enter into the Agreement or if Customer does not accept all Agreement terms and conditions, Customer should not register to use, deploy, configure, install, update, enable, manage, or use the Software Offerings.

Where this Agreement, Product Specific Terms, or Separate Component terms are presented or linked in connection with a download, access, installation, configuration, update, enablement, management, or deployment flow, including a scale deployment workflow, clicking or selecting the corresponding Download, Access, Accept, Deploy, or similar call-to-action, or proceeding with the deployment after such terms are presented, constitutes Customer’s acceptance of those terms.

You can find a short summary explaining how you can use certain Software Offerings here.

1. **LICENSE**

1.1 Grant. Subject to the terms of the Agreement, Customer’s Order Form and license parameters of an Enterprise Product Part Number, and payment of applicable fees, NVIDIA grants Customer a limited, worldwide, non-exclusive, non-transferable, non-sublicensable (except as expressly granted in the Agreement) license during the Term to install (including through centralized, automated, remote, or fleet-management deployment by an Administrator), use and reproduce the Software.

1.2 Limited Grants. The grants in Section 1.1 above are further limited as follows:

1.2.1 “not for resale” or “NFR” (typically a license to a distributor or reseller) licenses to Enterprise Products are licensed for limited use under Section 1.1 solely for internal evaluation or to demonstrate the Enterprise Product to others, but may not use, distribute or deploy in production.

1.2.2 trial licenses to Enterprise Products are licensed for limited use under Section 1.1 solely for the trial period.

1.3 Authorized Users.

1.3.1 Software Users. Under the Agreement, Customer Personnel may access and use the Software Offerings and Customer End Users may access and use Customer Products.

1.3.2 Customer Personnel. Customer is responsible for the compliance with the terms of the Agreement by Customer Personnel. Any act or omission that if committed by Customer would constitute a breach of the Agreement will be deemed to constitute a breach of the Agreement if committed by Customer Personnel.

1.3.3 Customer End Users. Customer must enter into enforceable agreements with Customer End Users, or maintain corporate policies applicable to Customer Personnel and other authorized users, that are at least as protective as the terms of the Agreement (including, but not limited to, terms relating to the grants, restrictions and ownership of intellectual property). Where Customer or an Administrator uses centralized, automated, remote, or fleet-management tools to deploy, configure, install, update, enable, or manage Software Offerings, Customer remains responsible for compliance with the Agreement by Customer Personnel and Customer End Users who access or use such Software Offerings, including where individual out-of-box or end-user license prompts are bypassed. NVIDIA reserves the right to, from time to time, update the Agreement (an updated version being referred to as “Updated Agreement”). Upon notification of an Updated Agreement, which may be in the form of electronic acceptance by Customer or an Administrator on Customer’s behalf of an Updated Agreement, Customer agrees to review the Updated Agreement and as appropriate update the terms it presents to Customer End Users and maintain compliance across the deployed fleet on a going forward basis. If NVIDIA or the Customer have reason to know or determine in good faith that a Customer End User is

not in compliance with the Agreement, NVIDIA and Customer will cooperate to investigate and resolve such non-compliance. Customer will cooperate to enforce use rights with Customer End Users.

1.4 License Types. The terms in this Section 1.4 apply only to Enterprise Products, not to Community Products. The license types below describe features that may be part of an Enterprise Product; and not all license types may be available for each Enterprise Product. Enterprise Products are licensed under the following license types: Subscription per CPU Socket, Subscription per GPU, Subscription per Node, Subscription per CCU, Subscription per HCA, Usage Based Subscription, or Perpetual license. Once Customer transitions from a Community Product to an Enterprise Product version under a paid Subscription or Perpetual license, any Customer use of such Enterprise Product must be under a paid Subscription or Perpetual license. Customer's order, license key or the product description will indicate the parameters of Customer's license. For Software Offerings deployed via centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that deployment parameters, including the number of GPUs, Nodes, Sockets, CCUs, HCAs, or other applicable license metrics, do not exceed the authorized license types and quantities specified in the applicable Order Form, license key, entitlement, or product description.

1.5 Replacement Products. NVIDIA may, from time to time at its discretion, give Customer the option to replace a certain Enterprise Product, subject to payment of applicable fees. In such cases, Customer must discontinue use of the replaced Enterprise Product timely upon the start of use of the replacement Enterprise Product. If requested in writing by NVIDIA, Customer will provide a written certificate signed by an authorized officer or Administrator affirming Customer's compliance with the replacement terms, including decommissioning of replaced Software across Customer's deployed fleet, as applicable.

1.6 Promotional Offerings. NVIDIA may, from time to time, offer free or discounted pricing programs covering certain uses of Software Offerings, for example having different license parameters or fees for evaluation or academic use. NVIDIA may stop accepting new sign-ups or discontinue a promotional offering at any time. Standard charges will apply after a promotional offering ends or if Customer exceeds the promotional offering use terms. Customer must comply with any additional terms, restrictions, or limitations (e.g., limitations on the total amount of usage) for a promotional offering as described in the corresponding offer terms.

2. **AI ETHICS.**

Use of the Software Offerings under the Agreement must be consistent with NVIDIA's Trustworthy AI terms at https://www.nvidia.com/en-us/agreements/trustworthy-ai/terms/. Where individual user acceptance is bypassed through centralized, automated, remote, or fleet-management deployment or configuration, Customer is responsible for ensuring that all Customer Personnel and Customer End Users use the Software Offerings consistent with those Trustworthy AI terms.

3. **PRE-RELEASE VERSIONS AND FEATURES**.

NVIDIA will clearly designate Software versions that are in Pre-Release. Pre-Release versions may not be fully functional, may contain errors or design flaws, and may have reduced or different security, privacy, accessibility, availability, and reliability standards relative to commercially provided NVIDIA software, materials and services. Use of a Pre-Release version may result in unexpected results, such as loss of use or loss of content. Customer may use a Pre-Release version at Customer's risk, understanding that such versions are not intended for use in business-critical systems and Customer may stop at Customer's convenience. Where Pre-Release versions are deployed, configured, installed, updated, enabled, or managed via centralized, automated, remote, or fleet-management tools, the Administrator acknowledges and accepts these risks on behalf of Customer and Customer is responsible for communicating and managing such risks for its authorized users. NVIDIA may choose to abandon development and terminate the availability of a Pre-Release version at any time without liability. Prerelease versions are provided "AS-IS," "WITH ALL FAULTS," and "AS-AVAILABLE," and are excluded from Enterprise Support.

4. **SERVICES.**

4.1 General Service Terms. Unless otherwise indicated by NVIDIA in this Agreement or an Order Form, Software Subscriptions include Enterprise Support, Maintenance and Updates. Enterprise Support, Maintenance and Updates may be sold separately for Perpetual licenses. After the expiration of Services, Customer retains the right to use a Perpetual license at the last-supported level subject to the terms of the Agreement. Where Customer or an Administrator deploys, configures, installs, updates, enables, or manages Software Offerings through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that only systems, devices, or Software Offerings with active and applicable Services, Maintenance, Update, Subscription, or support entitlements receive such Services, Maintenance, Updates, or support. Unless NVIDIA accepts Customer's request, which may be submitted by an Administrator on Customer's behalf, for Enterprise Support, Maintenance and Updates in an Order Form, NVIDIA is under no obligation to provide any Service. Unless Software is provided with their separate governing terms, they are deemed part of the applicable Software Offering and governed by the Agreement.

4.2 Use of Maintenance and Updates. NVIDIA encourages Customers to use Maintenance and Updates available to them. Customer's choice, including an Administrator's choice on Customer's behalf, not to deploy Maintenance or Update as they become available may result in issues with operability, compatibility and interoperability and result in the Software in use being non-conforming to later Software documentation.

4.3 Maintaining the Authorized Number of Licenses. Customer's use of Maintenance or Update does not change the number of authorized licenses. Customer agrees to promptly discontinue use of prior versions, including across Customer's deployed fleet, as necessary to maintain the authorized number of licenses.

4.4 Work Out of Scope. Any enhancements or additions to Software beyond Maintenance or Updates are outside of the scope of this Agreement.

4.5 Services for NFR, Trial or Developer Program Licenses. NVIDIA is not obligated to provide Services for any free NFR, trial or developer program items, even if identified as Subscription versions, and any Services are provided at NVIDIA's discretion. Customer's use of centralized, automated, remote, or fleet-management tools to distribute, deploy, configure, install, update, enable, or manage NFR, trial, or developer program items does not create any entitlement to Enterprise Support, Maintenance, Updates, or other Services.

4.6 Services to Customer End Users. As between NVIDIA and Customer, Customer is responsible for supporting Customer Products and for providing first-line support to Customer Personnel and Customer End Users for Software Offerings that Customer or an Administrator deploys, configures, installs, updates, enables, or manages.

5. **USERNAME AND PASSWORD.**

Customer is responsible for securely maintaining log-in information, including administrative credentials, API keys, service tokens, deployment credentials, and other credentials used for Customer Authorized Users' use or for centralized, automated, remote, or fleet-management deployment, configuration, installation, updating, enablement, or management of Software Offerings, and for all activities under Customer's account(s). Customer is responsible for ensuring that credentials used for automated provisioning are managed securely, limited to authorized personnel and workflows, and rotated or revoked as appropriate. Customer agrees to notify NVIDIA at enterprisesupport@nvidia.com immediately of any known or suspected security incidents or unauthorized use of Customer's account(s), credentials, or the Software Offering.

6. **COMPONENTS UNDER OTHER LICENSES.**

The Software may include or be distributed with Separate Components, including NVIDIA software development kits such as the CUDA Toolkit, TensorRT, cuDNN, NVIDIA drivers, and other components accompanied by separate legal notices or terms. The Separate Components are subject to the applicable OSS Licenses or other license terms, including any proprietary notices, disclaimers, require-

ments and extended use rights; except that the Agreement will prevail regarding the use of third-party open source software, unless NVIDIA components are provided under an OSS License or a third-party OSS License requires its license terms to prevail. Customer is responsible for maintaining all notices and requirements for Separate Components and for ensuring that Customer Personnel and Customer End Users receive or can access applicable Separate Component terms, including the NVIDIA CUDA Toolkit End User License Agreement available at https://docs.nvidia.com/cuda/eula/index.html, through a clear and conspicuous hyperlink or other retrievable notice in the relevant download, access, deployment, product, or documentation flow. For scale deployment workflows in which individual out-of-box or end-user license prompts are bypassed, Customer is responsible for ensuring that an Administrator has reviewed and accepted, on Customer's behalf, the Separate Component terms applicable to the Software Offerings being deployed, including the NVIDIA CUDA Toolkit End User License Agreement to the extent CUDA Toolkit components are included in or distributed with the Software.

7. **PAYMENT TERMS AND TAXES.**

7.1 Ordering. Customer may be able to purchase a Subscription or Perpetual license directly from NVIDIA, or via a reseller (in some cases a cloud marketplace reseller), as available. Each Order Form will be effective when entered into by the Customer and NVIDIA. Each order placed by Customer through an Order Form is a separate transaction of the parties under the Agreement.

7.2 Fees. When purchasing directly from NVIDIA, the following terms apply: Fees for the Subscriptions or Perpetual licenses are set forth in the associated Order Form and are payable pursuant to the terms of such Order Form. Unless otherwise expressly indicated in an Order Form, fees will be invoiced upon Customer's purchase, are payable upon invoice and are expressed in U.S. Dollars. Each Order Form placed is non-cancelable and fees received are non-refundable. All amounts not paid when due will accrue interest (without the requirement of a notice) at the lower of 1.5% per month or the highest rate permissible by law until the unpaid amounts are paid in full. Fees do not include any taxes, duties or similar charges.

7.3 Taxes. If NVIDIA is required to pay sales, use, property, value-added or other taxes based on the payments provided under the Agreement and if NVIDIA is required to collect and remit such taxes, then such taxes will be billed to and paid by Customer or Customer's reseller, unless NVIDIA receives a valid exemption or resale certificate. If Customer is not billed the applicable tax under the Order Form, then it is Customer's responsibility to properly remit the tax directly to the applicable tax jurisdiction. Further, Customer acknowledges that the payments to NVIDIA under the Agreement will be made in full without reduction for withholding taxes, if applicable. This section will not apply to taxes based on NVIDIA's net income or payroll taxes.

7.4 Overdue Payment. If any payment is overdue from Customer or a reseller, NVIDIA reserves the right to suspend the Subscriptions, Perpetual licenses and Services, in addition to any other remedies it may have, until the payment delinquency is corrected.

7.5 Price Changes. Any price change will only apply to purchases after the price change.

8. **LIMITATIONS.**

The following limitations and restrictions apply to the Software, Derivative Samples and Derivative Models, including any deployment scripts, images, configurations, automated workflows, and centralized or fleet-management processes used by or on behalf of Customer, and Customer is responsible for the consequences of non-conformance with these limitations:

8.1 Customer will use the Software exclusively for authorized purposes, consistent with the Agreement's terms and all applicable laws, regulations and the rights of others.

8.2 Product-Specific Terms may indicate that NVIDIA proprietary Software, Derivative Samples and Derivative Models are licensed only to run on NVIDIA Platforms. When deploying, configuring, installing, updating, enabling, or managing Software through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that such tools provision and maintain the

Software only on authorized NVIDIA Platforms or other systems expressly permitted by the applicable Product-Specific Terms.

8.3 Customer may not combine the use of paid and unpaid Software, Derivative Samples and Derivative Models in a way that avoids incurring fees or exceeding use limits or quotas.

8.4 Customer may not reverse engineer, decompile, disassemble Software components provided in binary form, nor attempt in any other manner to obtain source code of such Software components.

8.5 Except as expressly granted in the Agreement or Product-Specific Terms, Customer may not copy, sell, resell, rent, sublicense, transfer, assign, timeshare, distribute, modify, or create derivative works of any portion of the Software, including, without limitation, in any publicly accessible software repositories. Customer's internal use of centralized, automated, remote, or fleet-management tools to provision Software to Customer's authorized users is permitted only to the extent consistent with the Agreement and the applicable Product-Specific Terms, and Customer remains responsible for preventing unauthorized distribution, modification, or access to the Software.

8.6 Customer may not indicate that a product or service developed with the Software is sponsored or endorsed by NVIDIA unless expressly authorized in writing by NVIDIA.

8.7 Customer may not bypass, disable, or circumvent any technical limitation, encryption, security, digital rights management or authentication mechanism contained in the Software. Customer is responsible for ensuring that deployment scripts, images, configurations, automated workflows, and centralized or fleet-management tools do not bypass, disable, or circumvent such mechanisms, except to the extent NVIDIA expressly documents or authorizes a workflow for deployment or configuration.

8.8 Customer may not use the Software components governed by the Agreement in any manner that would cause components to become subject to an OSS License or other shareware license.

8.9 Customer may not distribute or disclose to third parties the results of benchmarking, competitive analysis, regression or performance data relating to the Software without the prior written permission from NVIDIA, except as described at https://docs.nvidia.com/nvidia-containers-benchmarking.pdf. Where Software is deployed, configured, installed, updated, enabled, or managed via centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that authorized users are notified of and comply with these benchmarking disclosure restrictions.

8.10 Customer may not replace any NVIDIA software components in the Software that are governed by the Agreement with other software that implements NVIDIA application programming interfaces (APIs). Customer is responsible for ensuring that deployment scripts, images, configurations, automated workflows, third-party management tools, and centralized or fleet-management tools do not substitute, replace, or remove such NVIDIA software components.

8.11 Customer may not reverse engineer, decompile or disassemble any portion of the output generated using an NVIDIA proprietary software development kit (e.g., NVIDIA CUDA toolkit), including their development tools and compilers. Where such outputs are generated, deployed, configured, managed, or processed via centralized, automated, remote, or fleet-management tools or workflows, Customer is responsible for ensuring that all authorized users and automated processes comply with these restrictions.

8.12 Customer may not use the Software or NVIDIA Confidential Information for the purpose of (i) developing competing products or technologies or assisting a third party in such activities, or (ii)identifying or supporting an assertion or potential assertion of any intellectual property rights against NVIDIA (including patent, copyright, or trade secret).

8.13 Customer acknowledges that the Software as delivered under the Agreement is not tested or certified by NVIDIA for use in any Critical Application. Beyond NVIDIA delivering the Software in accordance with the Agreement, NVIDIA will not be liable to Customer or any third party, in whole or in part, for any claims or damages arising from such uses. Customer is solely responsible for ensuring that systems and applications developed or deployed with the Software include sufficient safety and

redundancy features and comply with all applicable legal and regulatory standards and requirements. Where Software is deployed, configured, installed, updated, enabled, or managed at scale through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that target systems and use cases comply with this Section 8.13 across the deployed fleet.

8.14 Customer may not use the Software to infringe any third party's Intellectual Property Rights.

8.15 Customer agrees to defend, indemnify and hold harmless NVIDIA and its Affiliates, and their respective employees, contractors, agents, officers and directors, from and against any and all third-party claims, damages, obligations, losses, liabilities, costs or debt, fines, restitutions and expenses (including but not limited to attorney's fees and costs incident to establishing the right of indemnification) arising out of or related to (i) products or services developed or deployed with or that use the Software (including results or data generated from such use), or claims that such products or services violate laws, or infringe, violate, or misappropriate any third party right, provided that Customer is not obligated to provide intellectual property indemnification to NVIDIA for Software that Customer has not modified; or (ii) a violation of the terms and conditions of the Agreement. Where Software is deployed, configured, installed, updated, enabled, or managed via centralized, automated, remote, or fleet-management tools, Customer's indemnification obligations include activities, claims, and violations arising out of or related to such deployment, configuration, installation, updating, enablement, management, and subsequent use across Customer's deployed fleet.

9. **CONFIDENTIALITY.**

9.1 Right to Use Confidential Information. The receiving party may use the disclosing party's Confidential Information only to exercise its rights and perform its obligations under the Agreement. The receiving party may disclose the disclosing party's Confidential Information to its Affiliates, directors, officers, employees, contractors, independent external auditors, and advisors (including legal and accounting) who have a need to know that information in their work for the receiving party under the Agreement and who are bound to protect that information under obligations of confidentiality at least as restrictive as those described in the Agreement. The receiving party will not disclose the disclosing party's Confidential Information to any third party, except as expressly authorized in the Agreement, without first having written approval of the disclosing party. The receiving party will protect the Confidential Information of the disclosing party with at least the same degree of care that the receiving party uses to protect its own similar confidential and proprietary information, but no less than a reasonable degree of care. Irrespective of any expiration or termination of the Agreement, the receiving party's duty to protect the disclosing party's Confidential Information expires five (5) years after the date Confidential Information was disclosed. If the disclosing party sends a written request, the receiving party will promptly return or destroy all Confidential Information received from the disclosing party, together with copies, except that a party may retain archival copies in accordance with its document retention policy or as required by law.

9.2 Exceptions. Confidential Information does not and will not include information that: (i) was in the public domain at the time it was communicated to the receiving party by the disclosing party, or later entered the public domain other than by a breach of the Agreement by the receiving party, (ii) is or was rightfully received or known by the receiving party without restriction on disclosure or any obligation of confidentiality, (iii) is or was independently developed by or for the receiving party without using any of the disclosing party's Confidential Information, or (iv) is or was generally made available by the disclosing party without restriction on disclosure. The receiving party will not be liable for disclosure of Confidential Information required to be disclosed in response to a valid order by a court or other governmental body of competent jurisdiction. The receiving party will (unless restricted) provide the disclosing party with prior written notice of the required disclosure to permit the disclosing party to seek confidential treatment of such information and disclose only the information necessary to comply with the requirements.

10. **TERMINATION.**

10.1 Automatic License Termination. For Enterprise Products NVIDIA designates as Perpetual or Sub-

scription based, Customer license terminates automatically without notice from NVIDIA at the earlier of the expiration or termination of the applicable license or the Agreement ("Term"). Either party may terminate licenses for Community Products for convenience with thirty (30) days prior written notice to the other party.

10.2 Early Termination of the Agreement. NVIDIA may at any time with advance written notice terminate the Agreement or an Order Form (i) if use of a Software, Derivative Sample or Derivative Model materially breaches the terms of the Agreement, or violates the rights of others, or violates applicable laws and regulations, or (ii) as needed, in NVIDIA's reasonable judgment, to comply with laws and regulations. Customer may terminate at any time the Agreement or an Order Form if NVIDIA does not timely cure a material breach of the Agreement. If the termination basis is breach and the breach is curable, the other party will have thirty (30) days from the date of notification to cure the breach (or other period mutually agreed in writing if the breach is by a Customer End User and the breach was not known by the Customer), or five (5) days to cure if the breach is of a payment obligation.

10.3 Effects of Termination. Upon any expiration or termination of the Agreement or an Order Form, any fees owed to NVIDIA become immediately due and payable even if longer terms have been agreed earlier. NVIDIA or an NVIDIA reseller will also bill Customer for any outstanding payment obligations related to a Subscription or Perpetual license period in the event of early termination for any reason other than for NVIDIA's uncured material breach. Customer agrees to promptly discontinue use of the affected Software and any NVIDIA intellectual property contained in the Derivative Samples and Derivative Models, and destroy all copies in Customer's possession or control (including any licenses in use by Customer's Authorized Users). Upon written request, Customer will certify in writing that Customer has complied with Customer's commitments under this section.

10.4 Services Termination. Services end (i) at the earlier of the expiration or termination of a Software Subscription, and (ii) upon termination of a Perpetual license, and in either case no credit or refund will be provided.

10.5 Survival. Sections 7 through 17 of this Agreement will survive termination.

11. **DATA COLLECTION.**

11.1 Collection Purposes. Customer acknowledges that Software may collect data for the following purposes: (a) properly configure and optimize products for use with Software; (b) deliver content or service through the Software; (c) check for compliance with the license or detect fraud or other malicious activity; and (d) improve NVIDIA products and services. Information collected may include: (i) configuration data; (ii) operating system; (iii) installed applications and drivers used with Software; and (iv) application settings, performance and usage data. With Customer's consent, diagnostic data, including crash reports, may be collected. Further, NVIDIA may require certain personal information such as name, email address and entitlement information to deliver Software or provide Services to Customer. Please review documentation accompanying the relevant Software for data collection specific to the Software.

11.2 Third Party Privacy Practices. The Software may contain links to third party websites and services, and Customer's use of third-party products and services may be subject to such third-party privacy statements or practices. NVIDIA is not responsible for the privacy statements or practices of third parties.

11.3 Privacy Policy. The NVIDIA Privacy Policy, located at NVIDIA Privacy Policy explains NVIDIA's policy for collecting and using data, and Customer can visit the NVIDIA Privacy Center, located at Privacy Center | NVIDIA, to manage Customer's consent and privacy preferences. If Customer accesses the Software and Services through Customer's enterprise organization, please contact Customer's systems administrator with any questions relating to the collection and processing of Customer's data. If Customer is its enterprise organization's systems administrator, please contact enterprisesupport@nvidia.com to manage the organization's privacy preferences.

12. **OWNERSHIP, ATTRIBUTION AND FEEDBACK.**

12.1 NVIDIA Ownership. The Software Offerings, including all Intellectual Property Rights, are and will remain the sole and exclusive property of NVIDIA or its licensors. Except as expressly granted in the Agreement, (i) NVIDIA reserves all rights, interests and remedies in connection with the Software Offerings, and (ii) no other license or right is granted to Customer by implication, estoppel or otherwise.

12.2 Customer Ownership. Subject to the rights of NVIDIA and its suppliers in Software Offerings, Customer holds all rights, title and interest in and to Customer Products, Derivative Samples and Derivative Models including their respective Intellectual Property Rights.

12.3 Attribution. Customer may not change or remove copyright or other proprietary notices in the Software or misrepresent the authorship of the Software, and Customer must cause any modified files to carry prominent notices stating that Customer changed the files such that modifications are not misrepresented as an original Software. Except as necessary to give attribution to works as described in this section, Customer is not granted any trademark license under the Agreement.

12.4 Feedback. Customer may, but is not obligated to, provide Feedback to NVIDIA or an NVIDIA Affiliate. Feedback, even if designated as confidential by Customer, will not create any confidentiality obligation for NVIDIA or its Affiliates beyond not identifying Customer as the source of the Feedback. If Customer provides Feedback, Customer hereby grants NVIDIA, its Affiliates and its designees a non-exclusive, perpetual, irrevocable, sublicensable, worldwide, royalty-free, fully paid-up, and transferable license, under Customer's Intellectual Property Rights, to publicly perform, publicly display, reproduce, use, make, have made, sell, offer for sale, distribute (through multiple tiers of distribution), import, create derivative works of, and otherwise commercialize and exploit the Feedback for any purpose at NVIDIA's discretion. NVIDIA agrees that Customer Feedback is provided "AS-IS" without a warranty of any kind.

13. **WARRANTY DISCLAIMER.**

THE SOFTWARE OFFERINGS ARE PROVIDED BY NVIDIA AS-IS AND WITH ALL FAULTS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVIDIA DISCLAIMS ALL WARRANTIES AND REPRESENTATIONS OF ANY KIND, WHETHER EXPRESS, IMPLIED OR STATUTORY, RELATING TO OR ARISING UNDER THE AGREEMENT, INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF TITLE, NONINFRINGEMENT, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, USAGE OF TRADE AND COURSE OF DEALING. WITHOUT LIMITING THE FOREGOING, NVIDIA DOES NOT WARRANT THAT THE SOFTWARE OFFERINGS WILL MEET CUSTOMER'S REQUIREMENTS, THAT ANY DEFECTS OR ERRORS WILL BE CORRECTED. THAT ANY CERTAIN CONTENT WILL BE AVAILABLE, OR THAT SOFTWARE OFFERINGS ARE FREE OF VIRUSES OR OTHER HARMFUL COMPONENTS. NO INFORMATION OR ADVICE GIVEN BY NVIDIA WILL IN ANY WAY INCREASE THE SCOPE OF ANY WARRANTY EXPRESSLY PROVIDED IN THE AGREEMENT.

14. **LIMITATION OF LIABILITY.**

14.1 Disclaimers. TO THE MAXIMUM EXTENT PERMITTED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY (I) INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, OR (II) DAMAGES FOR THE (A) COST OF PROCURING SUBSTITUTIVE GOODS, OR (B) LOST PROFITS, REVENUE, USE, DATA OR GOODWILL ARISING OUT OF OR IN CONNECTION WITH THE AGREEMENT OR THE USE OR THE PERFORMANCE OF SOFTWARE OFFERINGS WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY, OR OTHERWISE, AND EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES AND EVEN IF A PARTY'S REMEDIES FAIL THEIR ESSENTIAL PURPOSE.

14.2 Damages Capped. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVIDIA'S TOTAL CUMULATIVE AGGREGATE LIABILITY FOR ANY AND ALL LIABILITIES, OBLIGATIONS OR CLAIMS ARISING OUT OF OR RELATED TO THE AGREEMENT WILL NOT EXCEED THE NET AMOUNT NVIDIA WAS PAID FOR THE SOFTWARE GIVING RISE TO THE CLAIM DURING THE TWELVE (12) MONTHS PERIOD BEFORE THE EVENT GIVING RISE TO THE LIABILITY (OR UP TO US$100.00 IF CUSTOMER OBTAINED SUCH SOFTWARE AT NO CHARGE).

15. **GOVERNING LAW.**

The Agreement will be governed in all respects by the laws of the United States and the laws of the State of Delaware, without regard to conflict of laws principles or the United Nations Convention on Contracts for the International Sale of Goods. The state and federal courts residing in Santa Clara County, California will have exclusive jurisdiction over any dispute or claim arising out of or related to the Agreement, and the parties irrevocably consent to personal jurisdiction and venue in those courts; except that either party may apply for injunctive remedies or an equivalent type of urgent legal relief in any jurisdiction.

16. **GENERAL.**

16.1 Assignment. NVIDIA may assign, delegate or transfer its rights or obligations under the Agreement by any means or operation of law. Customer agrees that Customer will not transfer or assign the Agreement or Customer's rights and obligations by any means or operation of law without NVIDIA's permission. Any attempted assignment not approved by NVIDIA in a signed writing will be void and of no effect.

16.2 Subcontracting. NVIDIA may subcontract obligations under the Agreement but will remain liable to Customer for any subcontracted obligations.

16.3 Notices. If NVIDIA needs to contact Customer about a Software Offering, Customer consents to receive notices electronically. Customer is responsible for keeping Customer's notification email current. The parties agree that electronic notice will satisfy any legal communication requirements, and that email notices will be treated as received when the email is sent. Customer is required to send legal notices to NVIDIA Corporation, 2788 San Tomas Expressway, Santa Clara, California 95051, United States of America, Attention: Legal Department, with an emailed copy to legalnotices@nvidia.com.

16.4 Trade and Compliance. Customer agrees to comply with all applicable export, import, trade and economic sanctions laws and regulations, as amended, including without limitation U.S. Export Administration Regulations and Office of Foreign Assets Control regulations. Customer confirms (a) its understanding that export or reexport of certain NVIDIA products or technologies may require a license or other approval from appropriate authorities and (b) that it will not export or reexport any products or technology, directly or indirectly, without first obtaining any required license or other approval from appropriate authorities, (i) to any countries that are subject to any U.S. or local export restrictions (currently including, but not necessarily limited to, Belarus, Cuba, Iran, North Korea, Russia, Syria, the Region of Crimea, Donetsk People's Republic Region and Luhansk People's Republic Region); (ii) to any end-user who it knows or has reason to know will utilize them in the design, development or production of nuclear, chemical or biological weapons, missiles, rocket systems, unmanned air vehicles capable of a maximum range of at least 300 kilometers, regardless of payload, or intended for military end-use, or any weapons of mass destruction; (iii) to any end-user who has been prohibited from participating in the U.S. or local export transactions by any governing authority; or (iv) to any known military or military-intelligence end-user or for any known military or military-intelligence end-use in accordance with U.S. trade compliance laws and regulations. Use of the Software Offerings under the Agreement must be consistent with NVIDIA's HumanRightsPolicy.pdf (nvidia.com).

16.5 Government Rights. The Software, including its documentation and technology ("Protected Items"), are "Commercial products" as this term is defined at 48 C.F.R. 2.101, consisting of "commercial computer software" and "commercial computer software documentation" as such terms are used in, respectively, 48 C.F.R. 12.212 and 48 C.F.R. 227.7202 & 252.227-7014(a)(1). Before any Protected Items are supplied to the U.S. Government, Customer will (i) inform the U.S. Government in writing that the Protected Items are and must be treated as commercial computer software and commercial computer software documentation developed at private expense; (ii) inform the U.S. Government that the Protected Items are provided subject to the terms of the Agreement; and (iii) mark the Protected Items as commercial computer software and commercial computer software documentation developed at private expense. In no event will Customer permit the U.S. Government to acquire rights in Protected Items beyond those specified in 48 C.F.R. 52.227-19(b)(1)-(2) or 252.227-7013(c) except as

expressly approved by NVIDIA in writing.

16.6 Force Majeure. Except for payment of fees or taxes, neither party will be liable during an instance of Force Majeure.

16.7 Audit. During the term of the Agreement and for a period of three (3) years thereafter, Customer will maintain complete and accurate information regarding Customer's activities under the Agreement. NVIDIA or an independent auditor will have the right to audit Customer to validate and confirm Customer's information and compliance with the terms of the Agreement. Audits will be conducted no more frequently than annually, unless non-compliance was previously found. If an audit reveals an underpayment, Customer will promptly remit the full amount of such underpayment to NVIDIA including interest that will accrue (without the requirement of a notice) at the lower of 1.5% per month or the highest rate permissible by law. If the underpaid amount exceeds five percent (5%) of the amounts payable to NVIDIA during the audited period or if the audit reveals a material non-conformance with the terms of the Agreement, then Customer will reimburse NVIDIA's reasonable audit costs. Further, Customer agrees that the party transacting to sell Software Offerings to Customer may share with NVIDIA information regarding Customer's compliance with the Agreement.

16.8 Entire Agreement. Regarding the subject matter of the Agreement, the parties agree that the Agreement constitutes the entire and exclusive agreement between the parties regarding the Software Offerings and supersedes all prior and contemporaneous communications. Any additional or different terms or conditions, whether contained in purchase orders, order acknowledgments, invoices or otherwise, will not be binding and are null and void.

16.9 Severability. If any provision of the Agreement is deemed invalid by a court of competent jurisdiction, the invalidity of such provision will not affect the validity of the remaining provisions of the Agreement, which will remain in full force and effect.

16.10 No Waiver. No waiver of any term of the Agreement will be deemed a further or continuing waiver of such term or any other term, and NVIDIA's failure to assert any right or provision under the Agreement will not constitute a waiver of such right or provision.

16.11 Independent Contractors. The parties are independent contractors, and this Agreement does not create a partnership, joint venture, joint employment, agency or other form of business association between the parties or their respective employees, or authorize either party to make or enter into any commitments for or on behalf of the other party. Neither party will have the power to bind the other party or incur any obligation on its behalf without the other party's prior written consent. Neither party, or their respective employees, is entitled to receive or make a claim for any benefits that the other party or its Affiliates may make available to its respective employees, such as health insurance, retirement plan contributions, equity awards or sick or vacation pay.

16.12 Independent Development. Nothing in the Agreement will be construed to limit or restrict either party from independently developing, providing, or acquiring any materials, services, products or technology that are similar to the subject of the Agreement, provided that the party does not breach its obligations under the Agreement in doing so.

16.13 Order of Precedence. If there is a conflict between the documents that make up the Agreement, the documents will control in the following order (starting with the highest order of precedence): (a) the Product Specific Terms, if applicable (b) the remainder of the Agreement (excluding the Product Specific Terms and URL Terms), (c) the URL Terms, and (d) the Order Form, if applicable.

16.14 Construction. The parties and their respective counsel have reviewed the Agreement and it will be interpreted fairly in accordance with its terms and without any strict construction in favor of or against either party. The headings in the Agreement are included solely for convenience and are not intended to affect the meaning or interpretation of the Agreement. As required by the context of the Agreement, the singular of a term includes the plural and vice versa.

17. **Definitions.**

17.1 “Affiliate” means an entity that owns or controls, is owned or controlled by, or is under common ownership or control with a party, where “control” is the possession, directly or indirectly, of the power to direct or cause the direction of the management and policies of an entity, whether through ownership of voting securities, by contract or otherwise.

17.1A “Administrator” means Customer Personnel or another individual authorized by Customer to deploy, configure, install, update, enable, manage, or otherwise administer Software Offerings on Customer systems or for Customer Personnel or Customer End Users.

17.2 “Authorized Users” means Customer Personnel and Customer End Users.

17.3 “CCU” means a license that allows concurrent Authorized Users to use the Software, however the simultaneous number of Authorized Users cannot exceed the maximum number of CCUs authorized under the license.

17.4 “Community Products” means Free SDKs and Supported SDKs.

17.5 “Compatible Application” means an application that includes the Software, while not adversely affecting the functionality of the other components in the Software.

17.6 “CPU Socket” means (i) for on-premise deployments, the number of physical processors in the computing environment on which the Enterprise Product is installed, or (ii) in a cloud computing environment, the compute instance on which the Enterprise Product is installed. For per CPU Socket licenses, NVIDIA requires one Enterprise Product license per compute instance.

17.7 “Confidential Information” means any non-public information provided, including the Order Form, in any format by one party to the other party under the Agreement during the term, that was designated or identified by the disclosing party as confidential or should be known by the receiving party when exercising reasonable business judgment to be confidential given the nature or circumstances of its disclosure.

17.8 “Critical Application” means any system or application where the use or failure of such system or application could result in injury, death, or catastrophic damage. Examples of Critical Applications include, without limitation, use in avionics, navigation, autonomous vehicle applications, AI solutions for automotive products, military, medical, or life support or other life or mission critical application.

17.9 “Customer End User” means any end user that receives access from Customer or its Affiliates to a Customer Product that includes, in whole or in part, the Software.

17.10 “Customer Infrastructure” means Customer owned or leased infrastructure.

17.11 “Customer Personnel” means employees and contractors of Customer and its Affiliates, and for educational institutions include enrolled students, who may access or use the Software Offerings from Customer Infrastructure to perform work authorized by the Agreement on Customer’s behalf.

17.12 “Customer Product” means a Customer product, which can be hosted or distributed, that (a) includes the Software (in whole or in part), (b) may include Derivative Samples and Derivative Models, and (c) includes value added functionality to Customer End Users such as services, hardware or other software.

17.13 “Derivative Models” means all modifications, derivatives, adaptations, extensions or enhancements to a Model, works based on a Model, or any other model created or initialized by transfer of patterns of the weights, parameters, activations or output of a Model, to the other model, to cause the other model to perform similarly to the Model, including (but not limited to) distillation methods entailing the use of intermediate data representations or methods based on the generation of synthetic data by the Model for training the other model.

17.14 “Derivative Samples” means all modifications, derivatives, adaptations, extensions or enhancements to sample or example source code delivered by NVIDIA that Customer develops as authorized under the Agreement.

17.15 “Enterprise Products” means the Software and Services that is provided via gated access to users who have a valid contract for the Subscription or Services, except as otherwise expressly permitted under the Agreement and the Product Specific Terms (as applicable).

17.16 “Enterprise Support” has the meaning assigned to it in the Enterprise Support Policy.

17.17 “Enterprise Support Policy” means the then-current support terms that apply to a particular Software Offering described at https://www.nvidia.com/en-us/agreements/enterprise-services/nvidia-enterprise-support-policy/.

17.18 “Error(s)” has the meaning assigned to it in the Enterprise Support Policy.

17.19 “Error Correction(s)” has the meaning assigned to it in the Enterprise Support Policy.

17.20 “Feedback” means suggestions, fixes, modifications, enhancements, techniques, requests, other feedback or materials, whether oral or in writing, for the Software Offerings.

17.21 “Force Majeure” means an event or circumstance that prevents or delays a party from performing its obligations under the Agreement and that event or circumstance: (i) is not within the reasonable control of that party and is not the result of that party’s negligence (including, without limitation, acts of God, natural disaster, acts of government, flood, fire, earthquakes, pandemics, civil unrest, war, acts of terror or general labor or delivery chain disturbances such as strikes), and (ii) cannot be overcome or avoided by that party using reasonably diligent efforts.

17.22 “Free SDKs” means the unsupported Software generally accessible to the public and free of charge, as updated from time to time, that Customer may use without a paid Subscription or Services.

17.23 “GPU” means (i) for on-premise deployments, the number of physical GPUs in the computing environment which is accessed by the Enterprise Product, or (ii) in a cloud computing environment, the number and type of GPUs attached to the compute instance on which the Enterprise Product is installed. For per GPU licenses, NVIDIA requires one Enterprise Product license for each GPU. For per GPU licenses that run on computing environments or compute instances without an NVIDIA GPU, NVIDIA requires one Enterprise Product license for each computing environment, or compute instance.

17.24 “HCA” means an NVIDIA host channel adapter in a Node.

17.25 “Intellectual Property Rights” means all intellectual property rights, including all patents, trademarks, trade dress, copyrights, database rights, trade secrets, know-how, mask works, and any other similar protected rights in any country including all related applications for and registrations of these rights.

17.26 “Maintenance” means security patch(es), documentation revisions, Error Correction(s) and Workaround(s) to the Supported Product made generally available by NVIDIA in its sole discretion to its other commercial customers of the Supported Product who have the same Supported Product version under a Service contract with NVIDIA that includes “Maintenance.”

17.27 “Model” means any Software that is a machine-learning based assembly (including checkpoints), consisting of learnt weights, parameters (including optimizer states) and configuration files that may be trained or tuned, in whole or in part, on data.

17.28 “NVIDIA Platform” means a system with NVIDIA GPUs or NVIDIA CPUs, or for networking a system with NVIDIA HCAs.

17.29 “Node” means (i) for on-premise deployments, the number of physical servers in the computing environment on which the Enterprise Product is installed, or (ii) in a cloud computing environment, the number of compute instances on which the Enterprise Product runs. For per Node licenses, NVIDIA requires one Enterprise Product license per Node.

17.30 “Order Form” means the order that Customer places with: (i) NVIDIA, or (ii) a cloud marketplace, in each case including all applicable sales terms and conditions issued by NVIDIA (such as a public list-

ing or private offer) and that are accepted by Customer for the purchase of Subscriptions or Perpetual licenses.

17.31 “OSS License” means any software, data or documentation subject to any license identified as an open source license by the Open Source Initiative (http://opensource.org), Free Software Foundation (http://www.fsf.org) or other similar open source organization or listed by the Software Package Data Exchange (SPDX) Workgroup under the Linux Foundation (http://www.spdx.org).

17.32 “Part Number” means an NVIDIA product identification for an Enterprise Product, consisting of a unique product code, product name and other characteristics about the product, such as the license type (e.g., per GPU) and the license duration.

17.33 “Perpetual” means a license to an Enterprise Product that is exclusive of Services, however NVIDIA may require Customer to separately purchase certain Services for a minimum period.

17.34 “Pre-Release” means a version or feature of the Software that is identified by NVIDIA as beta, developer preview, early access or otherwise as pre-release.

17.35 “Product Specific Terms” means the then-current terms available at https://www.nvidia.com/en-us/agreements/enterprise-software that apply to a particular Software Offering, which are incorporated into this Agreement by reference.

17.36 “Separate Component” means a component provided with separate legal notices or terms that accompany the components, such as OSS Licenses and other license terms.

17.37 “Services” means any combination of Enterprise Support, Maintenance and Updates.

17.38 “Software” means a Community Product or an Enterprise Product and their associated materials (such as NVIDIA AI Enterprise and NVIDIA Omniverse), all of which is available for on-premises use.

17.39 “Software Offering” means a specific Software together with Services, as applicable.

17.40 “Subscription” means Customer’s fixed duration Software license, inclusive of Services, unless otherwise expressly stated in this Agreement or an Order Form.

17.41 “Supported Products” has the meaning assigned to it in the Enterprise Support Policy.

17.42 “Supported SDKs” means Software generally accessible to the public and free of charge, as updated from time to time, that Customer may use without a paid Subscription or Services and that NVIDIA will provide support for as long as Customer has a valid paid Subscription to other Enterprise Products in the same product family. Examples of NVIDIA product families are NVIDIA AI Enterprise products and Omniverse products.

17.43 “Updates” means those modifications to the Supported Product other than Maintenance made generally available by NVIDIA in its sole discretion to its other commercial customers of the Supported Product who have the same Supported Product version under a Service contract with NVIDIA that includes “Updates” as indicated by a new version number (examples, from version 5.0 to version 5.2 or 6.0).

17.44 “URL Terms” are terms linked to in the Agreement, including Trustworthy AI Terms, NVIDIA’s privacy policy, the Enterprise Support Policy and NVIDIA Humans Rights Policy.

17.45 “Usage Based” means a license to an Enterprise Product that is metered and billed based on consumption, such as by the hour, and is inclusive of certain Enterprise Support for the duration of the license.

17.46 “Workaround” has the meaning assigned to it in the Enterprise Support Policy.

(v. May 5, 2025)

PRODUCT-SPECIFIC TERMS FOR NVIDIA AI PRODUCTS

These Product-Specific Terms for AI products ("AI Product-Specific Terms") are an exhibit to the NVIDIA Software License Agreement at https://www.nvidia.com/en-us/agreements/enterprise-software/nvidia-software-license-agreement/ (the "Agreement") and are incorporated as an integral part of the Agreement.

1. **Scope.** The Agreement and these AI Product-Specific Terms govern the use of NVIDIA AI products including:

1.1 Community Products, Free SDKs, the Software in the NVIDIA NGC AI catalog at https://ngc.nvidia.com/ (as updated from time to time) and publicly-available versions of NVIDIA Base Command Manager software and NVIDIA AI Workbench software, and

1.2 "DGX Systems" means NVIDIA-branded DGX systems, except for DGX Spark which may be NVIDIA or OEM branded systems.

1.3 Enterprise Products, the Software in NVIDIA AI Enterprise, NVIDIA Base Command Manager software, NVIDIA AI Workbench software, NVIDIA Aerial CUDA-Accelerated RAN, NVIDIA Run:ai, NVIDIA DGX Software (as defined) and NVIDIA Mission Control (as defined).

1.3 "NVIDIA AI Enterprise" means NVIDIA AI software in the NVIDIA NGC AI Enterprise catalog at https://ngc.nvidia.com/nvaie-supported, including NVIDIA NIMs and NVIDIA NeMo (except as otherwise expressly permitted under these Product Specific Terms), as updated from time to time.

1.4 "NVIDIA DGX Software" means NVIDIA-provided operating system and software specific for DGX Systems as indicated by NVIDIA in documentation, including the following if delivered: NVIDIA Base Command Manager, NVIDIA Magnum IO, NVIDIA Unified Fabric Manager, and other software and materials.

1.5 "NVIDIA Mission Control" means the NVIDIA provided software tool for management and monitoring NVIDIA GPU systems and includes features like cluster management, health monitoring and optimization.

1.6 "Research Projects" means the NVIDIA licensed source code released as part of the research projects listed at https://docs.nvidia.com/research-projects.pdf and documentation, and expressly exclude for the purpose of the Agreement any models, checkpoints, JSON file datasets, or third party licensed content available from the same repositories.

2. **Grant.** Section 1.1 of the Agreement is amended and restated to read as follows:

1.1 Subject to the terms of the Agreement, Customer's Order Form and license parameters of an Enterprise Product Part Number, and payment of applicable fees, NVIDIA grants Customer a limited, worldwide, non-exclusive, non-transferable, non-sublicensable (except as expressly provided in the Agreement) license to do the following for the duration of the license:

1.1.1 install, including through centralized, automated, remote, or fleet-management deployment by an Administrator, use, reproduce, create Derivative Samples and Derivative Models, and configure (if applicable) the Software,

1.1.2 offer as a service the Software, Derivative Samples and Derivative Models as part of a Customer Product, and

1.1.3 sublicense and distribute the Software, Derivative Samples and Derivative Models as part of a Customer Product.

3. **Limited Grant.** Section 1.2 of the Agreement is amended and restated to read as follows:

1.2 Limited Grants. The grants above are further limited as follows:

1.2.1 If the Customer Product is an application, it must be a Compatible Application.

1.2.2 NVIDIA DGX Software are only licensed for use in DGX Systems, and firmware licensed in or for specific DGX Systems may not be used in other NVIDIA or non-NVIDIA products. Where Customer

or an Administrator deploys, configures, installs, updates, enables, or manages NVIDIA DGX Software or such firmware through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that such tools provision and maintain the software and firmware only for permitted DGX Systems.

1.2.3 Base Command Manager when included in NVIDIA AI Enterprise only requires licensing for compute nodes under management, not management nodes.

1.2.4 NVIDIA AI Workbench is licensed for use under Section 1.1.1 and Section 1.1.3, not licensed for use under Section 1.1.2.

1.2.5 Customer may create a Custom TTS Application for their own use but cannot offer it as a service to third parties as described in Section 1.1.2 or distribute as part of a Compatible Application as described in Section 1.1.3. Customer must have sufficient rights and licenses for content used with a Custom TTS Application to generate new content. "Custom TTS Application" means a custom text-to-speech application that enables the creation of synthetic voice output that resembles an input voice.

1.2.6 "not for resale" or "NFR" (typically a license to a distributor or reseller) licenses to Enterprise Products are licensed for limited use under Section 1.1.1 solely for internal evaluation or to demonstrate the Enterprise Product to others.

1.2.7 trial licenses to Enterprise Products are licensed for limited use under Section 1.1 solely for the trial period.

1.2.8 the developer program for AI products provides access to Enterprise Products under licenses for personal use under Section 1.1.1 above for Software that is part of this program (as updated by NVIDIA from time to time) solely for internal evaluation, development or test non-production purposes. Software offered as part of the developer program is not for use, distribution or deployment in production. NVIDIA may admit persons into the developer program for AI products, as well as change or discontinue the program and its offerings, at NVIDIA's sole discretion.

1.2.9 Research Projects licenses to Enterprise Product Subscription for AI products may exercise only the grants in Section 1.1.1 and Section 1.1.2 above for Research Projects during the term of the Subscription solely for internal purposes, and Support, Maintenance and Updates are not provided for Software available as part of Research Projects. Customer will retrieve the Software from the Research Project repositories for use in accordance with the terms of the Agreement. Subject to the rights of NVIDIA and its suppliers in Software Offerings, Customer holds all rights, title and interest in and to Customer's output of Research Projects including their respective Intellectual Property Rights.

1.2.10 NVIDIA inference microservices (NVIDIA NIMs) designated by NVIDIA for use with NVIDIA RTX, GeForce RTX GPUs or NVIDIA Grace-Blackwell client systems as Free SDKs may be used without the purchase of a Subscription or Services if such NIMs are: (a) used on a PC or workstation with NVIDIA RTX or NVIDIA GeForce RTX GPUs and (b) not used in a commercial kiosk, server or other system used to service multiple users. For other uses of these NIMs, you may purchase an NVIDIA AI Enterprise Subscription or obtain a license under an NVIDIA program, as available.

4. **License Types.** Section 1.4 ("License Types") of the Agreement is amended to add the following:

NVIDIA may offer and sell, with different prices and terms, Subscriptions for use of Enterprise Products in a designated platform. For example, NVIDIA may indicate that a subset of the Software is supported in a certain platform (e.g., a Subscription for the IGX platform, DGX Spark, or RTX PRO Server) and offer and sell a dedicated Enterprise Product Subscription for use of the applicable Enterprise Product only in the identified platform. If you obtained a dedicated Enterprise Product Subscription at a different price solely for use in a particular platform and you use in other platforms, NVIDIA or a reseller may invoice you to pay a higher value Enterprise Product Subscription. Where Customer or an Administrator deploys, configures, installs, updates, enables, or manages platform-designated Subscriptions through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that the deployment remains limited to the designated platform and authorized

license quantities.

5. **A new subsection 1.7 is added to the Agreement as follows:**

1.7 Requirements. The requirements in this section apply to Software, Derivative Samples and Derivative Models to exercise the license grants in Sections 1.1.2 and 1.1.3:

1.7.3 NVIDIA Works Notice. The following notice must be included in source files of Derivative Samples and Derivative Models distributed: "This software contains source code provided by NVIDIA Corporation."

1.7.5 Use Report. Customer must report upon NVIDIA's email request, no more than monthly, the Software in use by Customer Personnel and Customer End Users Customer enabled, quantity, start and end dates, and any other reasonably requested information for NVIDIA to determine the fees due.

1.7.7 License Administration. Customer will be responsible for the administration of licenses and services to Customer's End Users during their use period, including where Customer or an Administrator provisions, deploys, configures, installs, updates, enables, or manages Software through centralized, automated, remote, or fleet-management tools.

6. **Limitation.** Section 8 of the Agreement is amended and restated to read as follows:

The following limitations and restrictions apply to the Software, Derivative Samples and Derivative Models, and Customer is responsible for the consequences of non-conformance with these limitations:

8.1 Customer will use the Software exclusively for authorized purposes, consistent with the Agreement's terms and all applicable laws, regulations and the rights of others.

8.2 Product-Specific Terms may indicate that NVIDIA proprietary Software, Derivative Samples and Derivative Models are licensed only to run on NVIDIA Platforms. When deploying, configuring, installing, updating, enabling, or managing Software through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that such tools provision and maintain the Software only on authorized NVIDIA Platforms or other systems expressly permitted by the applicable Product-Specific Terms.

8.3 Customer may not combine the use of paid and unpaid Software, Derivative Samples and Derivative Models in a way that avoids incurring fees or exceeding use limits or quotas.

8.4 Customer may not reverse engineer, decompile, disassemble Software components provided in binary form, nor attempt in any other manner to obtain source code of such Software components.

8.5 Except as expressly granted in the Agreement or Product-Specific Terms, Customer may not copy, sell, resell, rent, sublicense, transfer, assign, timeshare, distribute, modify, or create derivative works of any portion of the Software, including, without limitation, in any publicly accessible software repositories. Customer's internal use of centralized, automated, remote, or fleet-management tools to provision Software to Customer's authorized users is permitted only to the extent consistent with the Agreement and these Product-Specific Terms, and Customer remains responsible for preventing unauthorized distribution, modification, or access to the Software.

8.6 Customer may not indicate that a product or service developed with the Software is sponsored or endorsed by NVIDIA unless expressly authorized in writing by NVIDIA.

8.7 Customer may not bypass, disable, or circumvent any technical limitation, encryption, security, digital rights management or authentication mechanism contained in the Software. Customer is responsible for ensuring that deployment scripts, images, configurations, automated workflows, and centralized or fleet-management tools do not bypass, disable, or circumvent such mechanisms, except to the extent NVIDIA expressly documents or authorizes a workflow for deployment or configuration.

8.8 Customer may not use the Software components governed by the Agreement in any manner that would cause components to become subject to an OSS License or other shareware license.

8.9 Customer may not distribute or disclose to third parties the results of benchmarking, competitive analysis, regression or performance data relating to the Software without the prior written permission from NVIDIA, except as described at https://docs.nvidia.com/nvidia-containers-benchmarking.pdf. Where Software is deployed, configured, installed, updated, enabled, or managed via centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that authorized users are notified of and comply with these benchmarking disclosure restrictions.

8.10 Customer may not replace any NVIDIA software components in the Software that are governed by the Agreement with other software that implements NVIDIA application programming interfaces (APIs). Customer is responsible for ensuring that deployment scripts, images, configurations, automated workflows, third-party management tools, and centralized or fleet-management tools do not substitute, replace, or remove such NVIDIA software components.

8.11 Customer may not reverse engineer, decompile or disassemble any portion of the output generated using an NVIDIA proprietary software development kit (e.g., NVIDIA CUDA toolkit), including their development tools and compilers. Where such outputs are generated, deployed, configured, managed, or processed via centralized, automated, remote, or fleet-management tools or workflows, Customer is responsible for ensuring that all authorized users and automated processes comply with these restrictions.

8.12 Customer may not use the Software or NVIDIA Confidential Information for the purpose of (i) developing competing products or technologies or assisting a third party in such activities, or (ii)identifying or supporting an assertion or potential assertion of any intellectual property rights against NVIDIA (including patent, copyright, or trade secret).

8.13 Customer acknowledges that the Software as delivered under the Agreement is not tested or certified by NVIDIA for use in any Critical Application. Beyond NVIDIA delivering the Software in accordance with the Agreement, NVIDIA will not be liable to Customer or any third party, in whole or in part, for any claims or damages arising from such uses. Customer is solely responsible for ensuring that systems and applications developed or deployed with the Software include sufficient safety and redundancy features and comply with all applicable legal and regulatory standards and requirements. Where Software is deployed, configured, installed, updated, enabled, or managed at scale through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that target systems and use cases comply with this Section 8.13 across the deployed fleet.

8.14 Customer may not use the Software to infringe any third party's Intellectual Property Rights.

8.15 Customer agrees to defend, indemnify and hold harmless NVIDIA and its Affiliates, and their respective employees, contractors, agents, officers and directors, from and against any and all third-party claims, damages, obligations, losses, liabilities, costs or debt, fines, restitutions and expenses (including but not limited to attorney's fees and costs incident to establishing the right of indemnification) arising out of or related to (i) products or services developed or deployed with or that use the Software (including results or data generated from such use), or claims that such products or services violate laws, or infringe, violate, or misappropriate any third party right, provided that Customer is not obligated to provide intellectual property indemnification to NVIDIA for Software that Customer has not modified; or (ii) a violation of the terms and conditions of the Agreement. Where Software is deployed, configured, installed, updated, enabled, or managed via centralized, automated, remote, or fleet-management tools, Customer's indemnification obligations include activities, claims, and violations arising out of or related to such deployment, configuration, installation, updating, enablement, management, and subsequent use across Customer's deployed fleet.

8.16 The following proprietary Software components are licensed only to run on systems with NVIDIA Platforms: NVIDIA software development kits (such as CUDA toolkit, TensorRT, cuDNN) and NVIDIA drivers. When deploying, configuring, installing, updating, enabling, or managing Software through centralized, automated, remote, or fleet-management tools, Customer must configure such tools to ensure these components are only provisioned onto authorized NVIDIA Platforms.

8.17 Customer may not use the Software or any of its components for the purpose of emotion recog-

nition. Any technology included in the Software may only be used as fully integrated in the Software and consistent with all applicable documentation. Where Software is deployed, configured, installed, updated, enabled, or managed through centralized, automated, remote, or fleet-management tools, Customer is responsible for ensuring that authorized users and automated workflows comply with this restriction across the deployed fleet.

(v. June 13, 2025)

2.16. Third-Party License Notices

This NVIDIA product contains third party software that is being made available to you under their respective open source software licenses. Some of those licenses also require specific legal information to be included in the product. This section provides such information.

2.16.1. LIBICONV Library

NVTelemetry is using the `libiconv` library, which is licensed under the GNU Lesser General Public License (LGPL). `libiconv` source code can be obtained from the following link: https://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.15.tar.gz.

The `libiconv` library is provided under the following terms:

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
```

(continues on next page)

(continued from previous page)

```
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  1. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
```

(continues on next page)

(continued from previous page)

```
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  2. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.
```

(continues on next page)

(continued from previous page)

```
  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  3. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  4. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.
```

(continues on next page)

(continued from previous page)

```
  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  5. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  6. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    1.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
```

(continues on next page)

(continued from previous page)

```
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  7. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
```

(continues on next page)

(continued from previous page)

```
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.
```

(continues on next page)

(continued from previous page)

```
  8. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
```

(continues on next page)

(continued from previous page)

```
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  9. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  10. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.
```

(continues on next page)

(continued from previous page)

```
  11. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  12. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.
```

(continues on next page)

(continued from previous page)

```
  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  13. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
```

(continues on next page)

(continued from previous page)

```
License would be to refrain entirely from conveying the Program.

  14. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  15. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  16. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  17. Limitation of Liability.
```

(continues on next page)

(continued from previous page)

```
  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  18. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:
```

(continues on next page)

(continued from previous page)

```
    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>
```

2.17. Notices

2.17.1. Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

2.17.2. Trademarks

NVIDIA, the NVIDIA logo, DGX, DGX-1, DGX-2, DGX A100, DGX H100/H200, DGX B200, DGX Station, DGX Station A100, and DGX Spark are trademarks and/or registered trademarks of NVIDIA Corporation in the Unites States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

©2022-2026, NVIDIA Corporation