CDW Documentation

Datadog Agent Installation Guide for NVIDIA GPU Servers

Datadog Agent Installation Guide for NVIDIA GPU Servers

Infrastructure Monitoring and GPU Observability for NVIDIA DGX, HGX, and AI Workstation Platforms

Document Version: 2.0
Last Updated: May 2026
Target Platforms: NVIDIA DGX Spark, DGX H100, DGX A100, HGX H100, NVIDIA AI Workstations
Datadog Agent Version: 7.59.0+ (recommended)

Table of Contents

---

1. Overview

This guide covers installing the Datadog Agent on NVIDIA GPU servers running Ubuntu Linux. It applies to bare-metal servers, AI workstations, and DGX/HGX systems regardless of CPU architecture (x86-64 or ARM64). Once installed, the Datadog Agent provides:

- **Host observability** — CPU, memory, disk I/O, network throughput
- **GPU monitoring** — utilization, memory, temperature, power draw, NVLink bandwidth, and ECC error tracking via DCGM
- **Container monitoring** — all workloads running through the NVIDIA Container Runtime
- **Log aggregation** — system, driver, CUDA, and application logs
- **APM** — distributed tracing for AI inference services
- **Live process monitoring** — real-time view of GPU workload resource consumption

2. Supported NVIDIA Platforms

Platform	CPU Architecture	Typical OS	GPU Generation
DGX Spark	ARM64 (Grace Blackwell GB10)	Ubuntu 24.04 LTS	Blackwell
DGX H100	x86-64 (Intel Xeon)	Ubuntu 22.04 LTS	Hopper
DGX A100	x86-64 (AMD EPYC)	Ubuntu 20.04 / 22.04 LTS	Ampere
HGX H100 / H200	x86-64	Ubuntu 22.04 LTS	Hopper
NVIDIA AI Workstations	x86-64	Ubuntu 22.04 / 24.04 LTS	Ada / Blackwell

<note>The Datadog Agent supports both **x86-64** and **ARM64 (AArch64 / ARMv8)** on Ubuntu. The one-line installer auto-detects architecture and fetches the correct package.</note>

3. Prerequisites

3.1 Datadog Account Requirements

Requirement	Detail
Datadog Account	Active subscription (Free Trial, Pro, or Enterprise)
API Key	Found under **Organization Settings → API Keys**
Datadog Site	Know your region's intake URL (see table below)

Datadog Site Reference:

Region	Site Value
US1 (default)	`datadoghq.com`
US3	`us3.datadoghq.com`
US5	`us5.datadoghq.com`
EU	`datadoghq.eu`
AP1	`ap1.datadoghq.com`

3.2 System Requirements

Requirement	Minimum	Recommended
OS	Ubuntu 20.04 LTS	Ubuntu 22.04 or 24.04 LTS
Architecture	x86-64 or ARM64	—
Disk Space	1 GB free	3 GB (includes log buffers)
RAM for Agent	256 MB	512 MB
Datadog Agent	7.47.0+	7.59.0+
NVIDIA Driver	525+	Latest for your GPU generation
DCGM	3.1+	3.3+ (required for Blackwell / MIG)
Docker (optional)	20.10+	Latest

3.3 User and Permission Requirements

- A user account with `sudo` privileges
- The `dd-agent` system user is created automatically during installation — do not delete or modify it
- For GPU access: `dd-agent` must be added to the `video` and `render` groups (covered in Section 8)
- For Docker monitoring: `dd-agent` must be added to the `docker` group (covered in Section 11)

3.4 Software Dependencies

The one-line installer resolves these automatically:

```
curl
gnupg / gpg
apt-transport-https
lsb-release
ca-certificates
```

---

4. Pre-Installation Checklist

Work through this list before running the installer.

```
[ ] 1. Server is fully booted and reachable via SSH or terminal
[ ] 2. OS is Ubuntu 20.04, 22.04, or 24.04 LTS
[ ] 3. Internet connectivity confirmed: curl -I
https://install.datadoghq.com
[ ] 4. Datadog API key is ready
[ ] 5. Correct Datadog site URL is known
[ ] 6. System packages are up to date (see below)
[ ] 7. NVIDIA drivers are loaded: nvidia-smi returns GPU info
[ ] 8. Firewall allows outbound HTTPS (TCP 443) to Datadog endpoints
[ ] 9. No existing Datadog Agent is installed: dpkg -l | grep datadog
[ ]10. Disk has at least 1 GB free on /var and /etc
```

Verify GPU health before starting:

```
# Confirm GPU is detected and driver is loaded
nvidia-smi

# Check driver version
nvidia-smi --query-gpu=driver_version --format=csv,noheader

# Confirm CPU architecture
uname -m
# x86-64 machines output: x86_64
```

```
# DGX Spark / ARM machines output: aarch64

# Check OS version
lsb_release -a
```

Update the system:

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install -y curl gnupg apt-transport-https ca-certificates lsb-release
```

---

5. Network and Firewall Requirements

5.1 Required Outbound Endpoints

The Datadog Agent communicates outbound only over HTTPS (TCP 443). No inbound ports need to be opened on the server.

Endpoint	Port	Purpose
`*.datadoghq.com` (or your site)	443	Metrics, events, service checks
`agent-intake.logs.datadoghq.com`	443	Log ingestion
`trace.agent.datadoghq.com`	443	APM traces
`process.datadoghq.com`	443	Live process data
`install.datadoghq.com`	443	Installation script (one-time only)

5.2 Verify DNS Resolution

```
nslookup datadoghq.com
```

5.3 Proxy Configuration (If Required)

If your server routes through an HTTP proxy, set these environment variables before running the installer:

```
export http_proxy="http://proxy.yourcompany.com:8080"
export https_proxy="http://proxy.yourcompany.com:8080"
export no_proxy="localhost,127.0.0.1"
```

After installation, add proxy settings to the agent config (see Section 7.4).

5.4 Air-Gapped Environments

If the server has no internet access, download the correct package on a connected machine and

transfer it:

```
# For x86-64:
AGENT_VERSION="7.59.0"
curl -L -o datadog-agent_amd64.deb \
  "https://s3.amazonaws.com/apt.datadoghq.com/pool/d/da/datadog-
agent_${AGENT_VERSION}-1_amd64.deb"

# For ARM64 (DGX Spark, Grace-based servers):
curl -L -o datadog-agent_arm64.deb \
  "https://s3.amazonaws.com/apt.datadoghq.com/pool/d/da/datadog-
agent_${AGENT_VERSION}-1_arm64.deb"

# Transfer to target server:
scp datadog-agent_<arch>.deb user@<SERVER_IP>:/tmp/

# On the target server, install:
sudo dpkg -i /tmp/datadog-agent_<arch>.deb
```

<note>**Note:** `dpkg -i` installs only the package itself. It does **not** add the Datadog APT repository or GPG key. This means `apt upgrade` will not find future Agent updates. To enable managed updates, complete the steps below after the initial install.</note>

5.4.1 Adding the GPG Key and APT Repository (Post-dpkg Setup)

Run these commands on the air-gapped server to register the Datadog repository for future `apt`-based updates. The GPG keyring file must be transferred from a connected machine alongside the `.deb` package.

On the connected machine — download the GPG keyring:

```
curl -fsSL https://keys.datadoghq.com/DATADOG_APT_KEY_CURRENT.public \
  | gpg --dearmor -o datadog-keyring.gpg

# Transfer to target server alongside the .deb:
scp datadog-keyring.gpg user@<SERVER_IP>:/tmp/
```

On the air-gapped server — install the key and add the repository:

```
# Install the GPG keyring
sudo mv /tmp/datadog-keyring.gpg /usr/share/keyrings/datadog-archive-
keyring.gpg
sudo chmod 644 /usr/share/keyrings/datadog-archive-keyring.gpg

# Add the APT repository entry
echo "deb [signed-by=/usr/share/keyrings/datadog-archive-keyring.gpg] \
  https://apt.datadoghq.com/ stable 7" \
  | sudo tee /etc/apt/sources.list.d/datadog.list
```

Future updates (once connectivity is available or via an internal mirror) will be handled by `sudo apt update && sudo apt install datadog-agent`.

5.4.2 Post-Install Configuration (datadog.yaml)

When using `dpkg -i` directly, the installer script does **not** auto-populate `/etc/datadog-agent/datadog.yaml`. You must configure it manually before starting the Agent.

```
# Copy the example config as the working config
sudo cp /etc/datadog-agent/datadog.yaml.example /etc/datadog-agent/datadog.yaml

# Set your API key
sudo sed -i 's/# api_key:.*/api_key: <YOUR_DD_API_KEY>/' /etc/datadog-agent/datadog.yaml

# Set your Datadog site (default is datadoghq.com)
sudo sed -i 's/# site:.*/site: datadoghq.com/' /etc/datadog-agent/datadog.yaml

# (Optional) Set a custom hostname
sudo sed -i 's/# hostname:.*/hostname: <YOUR_HOSTNAME>/' /etc/datadog-agent/datadog.yaml

# Set correct ownership
sudo chown dd-agent:dd-agent /etc/datadog-agent/datadog.yaml

# Enable and start the Agent
sudo systemctl enable datadog-agent
sudo systemctl start datadog-agent
```

Verify the Agent is reporting correctly:

```
sudo datadog-agent status
```

5.4.3 Recovery from Failed Installation

If the installation partially fails or the Agent is misconfigured and non-functional, follow these steps to restore a clean state before retrying.

Step 1 — Stop and disable the Agent service (if it started):

```
sudo systemctl stop datadog-agent
sudo systemctl disable datadog-agent
```

Step 2 — Remove the package while preserving config (for diagnosis):

```
sudo dpkg -r datadog-agent
```

Or, for a full purge including config files:

```
sudo dpkg --purge datadog-agent
```

Step 3 — Remove residual files left outside the package manifest:

```
sudo rm -rf /etc/datadog-agent /var/log/datadog /opt/datadog-agent
/run/datadog-agent
sudo userdel dd-agent 2>/dev/null; sudo groupdel dd-agent 2>/dev/null
```

Step 4 — Clear the APT state (if the repository was partially added):

```
sudo rm -f /etc/apt/sources.list.d/datadog.list
sudo rm -f /usr/share/keyrings/datadog-archive-keyring.gpg
sudo apt update
```

Step 5 — Reinstall. Return to section 5.4 and repeat the download-and-install steps, or use the one-line installer (section 6) if internet access has become available.

<note tip>Before purging, copy `/etc/datadog-agent/` to a backup location. If the failure was a config issue rather than a package issue, you may be able to restore the backup config after a clean reinstall rather than reconstructing it from scratch.</note>

---

6. Installation

One-Line Installer (Official Method)

This is the only installation method covered in this guide. It is the official, fully automated method recommended by Datadog. The script auto-detects your OS version and CPU architecture, adds the Datadog APT repository with a verified GPG key, installs Agent 7, writes the configuration file with your API key, and enables the systemd service.

```
DD_API_KEY="<YOUR_DD_API_KEY>" \
DD_SITE="datadoghq.com" \
bash -c "$(curl -L
https://install.datadoghq.com/scripts/install_script_agent7.sh)"
```

Replace:

- <YOUR_DD_API_KEY> with your actual key from Datadog's Organization Settings
- `datadoghq.com` with your regional site URL if different

Optional variables you can prepend to customize the installation:

Variable	Example	Purpose
DD_HOSTNAME	dgx-h100-node-01	Sets a custom hostname visible in Datadog

Variable	Example	Purpose
DD_TAGS	“env:prod,team:mlops,gpu:h100”	Applies host-level tags to all telemetry immediately
DD_AGENT_MAJOR_VERSION	7	Pins to Agent v7 (already the default)

Full example with all options:

```
DD_API_KEY="<YOUR_DD_API_KEY>" \
DD_SITE="datadoghq.com" \
DD_HOSTNAME="nvidia-gpu-server-01" \
DD_TAGS="env:production,team:mlops,hardware:nvidia-dgx,gpu_arch:hopper" \
DD_AGENT_MAJOR_VERSION=7 \
bash -c "$(curl -L
https://install.datadoghq.com/scripts/install_script_agent7.sh)"
```

What the installer does, step by step:

1. Downloads and verifies the Datadog GPG signing key
2. Adds `apt.datadoghq.com` to your APT sources list
3. Runs `apt-get install datadog-agent`
4. Writes `/etc/datadog-agent/datadog.yaml` with your API key pre-populated
5. Enables and starts the `datadog-agent` systemd service

Verify the installation completed successfully:

```
# Check service is running
sudo systemctl status datadog-agent

# Run the built-in status check
sudo datadog-agent status

# Confirm version and architecture
sudo datadog-agent version
```

Expected output from `datadog-agent version`:

```
Agent 7.59.0 - Commit: xxxxxxx
Go Version: go1.22.x
Python Version: 3.12.x
```

---

7. Core Agent Configuration

The main configuration file is:

```
/etc/datadog-agent/datadog.yaml
```

All integration check configs live in:

```
/etc/datadog-agent/conf.d/<integration>.d/conf.yaml
```

Open the main config for editing:

```
sudo nano /etc/datadog-agent/datadog.yaml
```

7.1 Essential Configuration

/etc/datadog-agent/datadog.yaml

```
###################################
## Core Identity
###################################

# Your Datadog API key
api_key: <YOUR_DD_API_KEY>

# Datadog intake site — must match your account region
site: datadoghq.com

# Unique hostname for this server as it appears in Datadog
hostname: nvidia-gpu-server-01

# Host-level tags applied to ALL metrics, logs, traces, and events
tags:
  - env:production
  - hardware:nvidia-gpu-server
  - team:mlops

###################################
## Logging
###################################

log_level: info
log_file: /var/log/datadog/agent.log

###################################
## Collection Interval
###################################

min_collection_interval: 15

###################################
## Features
###################################

process_config:
  process_collection:
    enabled: true
```

```
logs_enabled: true

logs_config:
  container_collect_all: true

network_config:
  enabled: true

###################################
## APM
###################################

apm_config:
  enabled: true
  receiver_host: 0.0.0.0
  receiver_port: 8126
```

7.2 Apply Configuration Changes

Any time you edit `datadog.yaml` or a check config, restart the agent:

```
sudo systemctl restart datadog-agent
sudo datadog-agent status
```

7.3 Validate Connectivity

```
sudo datadog-agent diagnose connectivity-datadog
```

7.4 Proxy Configuration (If Required)

Add this section to `datadog.yaml` if your server routes through a proxy:

```
proxy:
  http: http://proxy.yourcompany.com:8080
  https: http://proxy.yourcompany.com:8080
  no_proxy:
    - localhost
    - 127.0.0.1
    - 169.254.0.0/16
```

---

8. GPU Monitoring Setup (DCGM)

NVIDIA DCGM (Data Center GPU Manager) is the recommended and most comprehensive method for GPU telemetry. It provides richer metrics than any alternative, including MIG (Multi-Instance GPU) support, ECC error tracking, NVLink bandwidth, XID error monitoring, and per-process GPU utilization. The DCGM check has been bundled in the Datadog Agent since v7.47.0.

8.1 Install and Start DCGM on the Host

Most NVIDIA DGX and HGX systems ship with DCGM pre-installed. Verify this first:

```
dcgmi --version
```

If DCGM is not present, install it:

```
sudo apt-get install -y datacenter-gpu-manager

sudo systemctl enable nvidia-dcgm
sudo systemctl start nvidia-dcgm

# Verify DCGM can see the GPUs
dcgmi discovery -l
```

<note important> **DCGM Version Requirements:**

- Ampere (A100) and older: DCGM 3.1+
- Hopper (H100): DCGM 3.2+
- Blackwell (DGX Spark, B100): DCGM 3.3+

</note>

8.2 Run the DCGM Exporter Container

The DCGM Exporter is a lightweight container that exposes GPU metrics as a Prometheus-compatible HTTP endpoint. The Datadog Agent scrapes this endpoint.

```
# Pull the DCGM Exporter image
docker pull nvcr.io/nvidia/k8s/dcgm-exporter:3.3.0-3.2.0-ubuntu22.04

# Run the exporter
docker run -d \
  --name dcgm-exporter \
  --restart=always \
  --gpus all \
  --cap-add SYS_ADMIN \
  -p 9400:9400 \
  -e DCGM_EXPORTER_LISTEN=":9400" \
  -e DCGM_EXPORTER_KUBERNETES="false" \
  nvcr.io/nvidia/k8s/dcgm-exporter:3.3.0-3.2.0-ubuntu22.04
```

Verify the exporter is serving metrics:

```
curl http://localhost:9400/metrics | head -30
```

You should see output like:

```
DCGM_FI_DEV_GPU_UTIL{gpu="0",UUID="...",device="nvidia0"} 42
DCGM_FI_DEV_FB_USED{gpu="0",...} 8192
DCGM_FI_DEV_POWER_USAGE{gpu="0",...} 275.5
```

8.3 Grant dd-agent Access to GPU Devices

```
sudo usermod -aG video dd-agent
sudo usermod -aG render dd-agent
```

8.4 Configure the Datadog DCGM Check

```
sudo mkdir -p /etc/datadog-agent/conf.d/dcgm.d/
sudo nano /etc/datadog-agent/conf.d/dcgm.d/conf.yaml
```

/etc/datadog-agent/conf.d/dcgm.d/conf.yaml

```
init_config:

instances:
  - prometheus_url: http://localhost:9400/metrics
    namespace: dcgm
    metrics:

      # Utilization
      - DCGM_FI_DEV_GPU_UTIL: gpu.utilization
      - DCGM_FI_DEV_MEM_COPY_UTIL: gpu.mem_copy_util

      # Memory
      - DCGM_FI_DEV_FB_FREE: gpu.framebuffer.free
      - DCGM_FI_DEV_FB_USED: gpu.framebuffer.used
      - DCGM_FI_DEV_FB_TOTAL: gpu.framebuffer.total

      # Thermals and Power
      - DCGM_FI_DEV_GPU_TEMP: gpu.temperature
      - DCGM_FI_DEV_MEMORY_TEMP: gpu.memory_temperature
      - DCGM_FI_DEV_POWER_USAGE: gpu.power_usage
      - DCGM_FI_DEV_TOTAL_ENERGY_CONSUMPTION: gpu.energy_consumption

      # Clock Speeds
      - DCGM_FI_DEV_SM_CLOCK: gpu.sm_clock
      - DCGM_FI_DEV_MEM_CLOCK: gpu.mem_clock
```

```
    # PCIe and NVLink Bandwidth
    - DCGM_FI_DEV_PCIE_TX_THROUGHPUT: gpu.pcie_tx
    - DCGM_FI_DEV_PCIE_RX_THROUGHPUT: gpu.pcie_rx
    - DCGM_FI_DEV_NVLINK_BANDWIDTH_TOTAL: gpu.nvlink_bandwidth

    # ECC Errors
    - DCGM_FI_DEV_ECC_SBE_VOL_TOTAL: gpu.ecc.single_bit_errors
    - DCGM_FI_DEV_ECC_DBE_VOL_TOTAL: gpu.ecc.double_bit_errors
    - DCGM_FI_DEV_RETIRED_SBE: gpu.retired_pages.sbe
    - DCGM_FI_DEV_RETIRED_DBE: gpu.retired_pages.dbe

    # XID Errors (critical GPU health signal)
    - DCGM_FI_DEV_XID_ERRORS: gpu.xid_errors

  tags:
    - "integration:dcgm"
    - "platform:nvidia-gpu-server"
```

<note tip>**Performance note:** `DCGM_FI_DEV_GPU_UTIL` can cause elevated collection overhead on some systems. If you observe high agent CPU usage, remove it and rely on `DCGM_FI_DEV_MEM_COPY_UTIL` and SM clock metrics as proxy indicators.</note>

8.5 Restart and Verify

```
sudo systemctl restart datadog-agent

# Run the DCGM check manually to see collected metrics
sudo datadog-agent check dcgm
```

8.6 Metrics Reference

Datadog Metric	Description
`dcgm.gpu.utilization`	GPU compute utilization (%)
`dcgm.gpu.mem_copy_util`	Memory copy engine utilization (%)
`dcgm.gpu.framebuffer.used`	GPU VRAM used (MB)
`dcgm.gpu.framebuffer.free`	GPU VRAM free (MB)
`dcgm.gpu.temperature`	GPU die temperature (°C)
`dcgm.gpu.memory_temperature`	HBM memory temperature (°C)
`dcgm.gpu.power_usage`	Current power draw (W)
`dcgm.gpu.energy_consumption`	Total energy consumed (mJ)
`dcgm.gpu.sm_clock`	Streaming Multiprocessor clock (MHz)
`dcgm.gpu.mem_clock`	Memory clock (MHz)
`dcgm.gpu.pcie_tx`	PCIe transmit throughput (KB/s)
`dcgm.gpu.pcie_rx`	PCIe receive throughput (KB/s)
`dcgm.gpu.nvlink_bandwidth`	NVLink total bandwidth (KB/s)
`dcgm.gpu.ecc.single_bit_errors`	Single-bit ECC errors (volatile)

Datadog Metric	Description
dcgm.gpu.ecc.double_bit_errors	Double-bit ECC errors (volatile)
dcgm.gpu.xid_errors	XID error count (GPU hardware errors)

9. Log Collection

9.1 Enable Log Collection

Confirm this is set in /etc/datadog-agent/datadog.yaml:

```
logs_enabled: true
```

9.2 Collect System and GPU Logs

```
sudo mkdir -p /etc/datadog-agent/conf.d/nvidia.d/
sudo nano /etc/datadog-agent/conf.d/nvidia.d/conf.yaml
```

/etc/datadog-agent/conf.d/nvidia.d/conf.yaml

```
logs:
  # Kernel and GPU messages (XID errors appear here)
  - type: file
    path: /var/log/syslog
    source: syslog
    service: nvidia-gpu-server

  # NVIDIA driver installer log
  - type: file
    path: /var/log/nvidia-installer.log
    source: nvidia-driver
    service: nvidia-gpu-server

  # DCGM logs
  - type: file
    path: /var/log/dcgm/*.log
    source: dcgm
    service: nvidia-dcgm

  # System journal (covers all services)
  - type: journald
    source: journald
    service: nvidia-gpu-server
```

9.3 Grant dd-agent Log File Access

```
sudo setfacl -m u:dd-agent:r /var/log/syslog
sudo setfacl -m u:dd-agent:r /var/log/nvidia-installer.log
sudo setfacl -Rm u:dd-agent:rX /var/log/dcgm/
```

---

10. Process and Container Monitoring

10.1 Live Process Monitoring

In `/etc/datadog-agent/datadog.yaml`:

```
process_config:
  process_collection:
    enabled: true
  # Scrub sensitive values (passwords, tokens) from process arguments
  scrub_args: true
  custom_sensitive_words:
    - "api_key"
    - "token"
    - "secret"
    - "password"
```

10.2 Network Performance Monitoring

NPM is supported on Ubuntu 20.04+ with kernel 4.14+. NVIDIA GPU servers typically run kernels well above this threshold.

```
network_config:
  enabled: true
```

```
# Load the required kernel module
sudo modprobe nf_conntrack
sudo systemctl restart datadog-agent
```

---

11. Docker and NVIDIA Container Runtime Integration

NVIDIA GPU servers run AI workloads in Docker containers using the NVIDIA Container Runtime. The following steps give the Datadog Agent visibility into those containers.

11.1 Add dd-agent to the Docker Group

```
sudo usermod -aG docker dd-agent
sudo systemctl restart datadog-agent
```

11.2 Configure the Docker Check

```
sudo mkdir -p /etc/datadog-agent/conf.d/docker.d/
sudo nano /etc/datadog-agent/conf.d/docker.d/conf.yaml
```

/etc/datadog-agent/conf.d/docker.d/conf.yaml

```
init_config:

instances:
  - url: "unix:///var/run/docker.sock"
    collect_container_size: true
    collect_images_stats: true
    collect_image_size: true
    collect_disk_stats: true
    collect_events: true
    tags:
      - "runtime:nvidia-container-runtime"
```

11.3 Auto-Collect Logs from All GPU Containers

In `/etc/datadog-agent/datadog.yaml`:

```
logs_config:
  container_collect_all: true
```

This captures stdout/stderr from every Docker container running on the server, including AI frameworks, inference servers, and training jobs.

---

12. Verification and Validation

12.1 Full Agent Status Report

```
sudo datadog-agent status
```

This outputs a complete breakdown of every running check, last collection time, any errors, and the number of metrics emitted.

12.2 Run Individual Checks Manually

```
# GPU check
sudo datadog-agent check dcgm

# Docker check
sudo datadog-agent check docker

# Core system checks
sudo datadog-agent check cpu
sudo datadog-agent check memory
sudo datadog-agent check disk
sudo datadog-agent check network
```

12.3 Full Connectivity Diagnostics

```
sudo datadog-agent diagnose connectivity-datadog

# Run all available diagnostics
sudo datadog-agent diagnose all
```

12.4 Confirm Data Is Arriving in the Datadog UI

1. Log into your Datadog account
2. Go to **Infrastructure → Infrastructure List**
3. Search for your server hostname (e.g., `nvidia-gpu-server-01`)
4. Click the host — CPU, memory, disk, and network metrics should be visible
5. Go to **Metrics → Explorer**, query `dcgm.gpu.utilization` to confirm GPU metrics are flowing
6. Go to **Logs** and filter by `source:dcgm` or `source:nvidia-driver` to confirm logs are arriving

12.5 Monitor Agent Logs in Real-Time

```
sudo tail -f /var/log/datadog/agent.log
sudo journalctl -u datadog-agent -f
```

---

13. Agent Management Commands

Command	Description
`sudo systemctl start datadog-agent`	Start the agent
`sudo systemctl stop datadog-agent`	Stop the agent
`sudo systemctl restart datadog-agent`	Restart the agent (required after config changes)
`sudo systemctl status datadog-agent`	Show current service status
`sudo systemctl enable datadog-agent`	Enable auto-start on boot

Command	Description
`sudo systemctl disable datadog-agent`	Disable auto-start on boot
`sudo datadog-agent status`	Show full agent status with all checks
`sudo datadog-agent version`	Show agent version and build info
`sudo datadog-agent check <name>`	Run a specific check and print output
`sudo datadog-agent configcheck`	Validate configuration files for syntax errors
`sudo datadog-agent diagnose all`	Run all built-in diagnostics
`sudo datadog-agent flare`	Bundle logs and config into a support archive
`sudo datadog-agent hostname`	Show the resolved hostname sent to Datadog
`sudo datadog-agent health`	Quick health check of the agent process

14. Troubleshooting

14.1 Agent Fails to Start

```
# Check for YAML syntax errors in config
sudo datadog-agent configcheck

# View recent service logs
sudo journalctl -u datadog-agent --since "10 minutes ago" --no-pager

# Manually validate YAML syntax
python3 -c "import yaml; yaml.safe_load(open('/etc/datadog-agent/datadog.yaml'))"
```

14.2 GPU Metrics Not Appearing

```
# Confirm the DCGM Exporter is running and exposing metrics
docker ps | grep dcgm-exporter
curl http://localhost:9400/metrics | head -10

# If exporter is running but metrics are missing in Datadog, run the check manually
sudo datadog-agent check dcgm

# Check GPU device access for dd-agent
sudo -u dd-agent nvidia-smi
# If this fails, add dd-agent to video/render groups:
sudo usermod -aG video dd-agent
sudo usermod -aG render dd-agent
sudo systemctl restart datadog-agent
```

14.3 DCGM Exporter Container Fails to Start

```
# Check container startup errors
docker logs dcgm-exporter

# Verify NVIDIA Container Runtime is functional
docker run --rm --gpus all nvidia/cuda:12.0-base-ubuntu22.04 nvidia-smi

# Verify DCGM service is running on the host
sudo systemctl status nvidia-dcgm
```

14.4 Logs Not Being Collected

```
# Confirm log collection is enabled
sudo datadog-agent status | grep -A 10 "Logs Agent"

# Check file permissions for dd-agent
getfacl /var/log/syslog

# Run the journald check in debug mode
sudo datadog-agent check journald -l debug
```

14.5 Docker Metrics Not Appearing

```
# Verify dd-agent is in the docker group
groups dd-agent

# Test Docker socket access as dd-agent
sudo -u dd-agent docker ps

# If permission denied, re-run:
sudo usermod -aG docker dd-agent
sudo systemctl restart datadog-agent
```

14.6 High Agent CPU Usage

```
# Identify which check is the cause
sudo datadog-agent status | grep -A 5 "Check Statistic"

# Reduce frequency for heavy checks by adding to the relevant conf.yaml:
# min_collection_interval: 60

# For DCGM specifically, comment out DCGM_FI_DEV_GPU_UTIL if it is the cause
```

14.7 Ubuntu 24.04 Integration Install Failure (PEP 668)

On Ubuntu 24.04, installing additional integrations via `pip` may fail with “externally managed

environment”:

```
sudo -u dd-agent /opt/datadog-agent/embedded/bin/pip install \
  datadog-nvml \
  --break-system-packages \
  --target /opt/datadog-agent/embedded/lib/python3.12/site-packages/
```

---

15. Security Hardening

15.1 Restrict Config File Permissions

The `datadog.yaml` file contains your API key and must not be world-readable:

```
sudo chmod 640 /etc/datadog-agent/datadog.yaml
sudo chown dd-agent:dd-agent /etc/datadog-agent/datadog.yaml

# Verify
ls -la /etc/datadog-agent/datadog.yaml
# Expected: -rw-r----- 1 dd-agent dd-agent
```

15.2 Use a Secrets Manager for the API Key

Instead of storing the API key in plain text in `datadog.yaml`, use the Datadog Secrets Backend to pull it from HashiCorp Vault, AWS Secrets Manager, or any secrets provider.

In `datadog.yaml`:

```
api_key: ENC[<your_secret_handle>]

secret_backend_command: /path/to/your/secrets-helper-script.sh
```

15.3 Rotate API Keys Regularly

In the Datadog UI: **Organization Settings → API Keys** — create a new key, update `datadog.yaml`, restart the agent, then revoke the old key.

15.4 Scrub Sensitive Process Arguments

Ensure `scrub_args: true` is set in `datadog.yaml` (see Section 10.1) to prevent credentials from appearing in Live Process monitoring.

15.5 Principle of Least Privilege

The `dd-agent` user should only belong to the groups it strictly needs:

```
# Required groups:
# video, render  — GPU device access
# docker         — only if Docker monitoring is enabled

# Verify group membership
groups dd-agent
```

Do not grant `dd-agent` sudo rights or add it to privileged groups beyond what is listed above.

---

16. Uninstallation

```
# Stop and disable the service
sudo systemctl stop datadog-agent
sudo systemctl disable datadog-agent

# Purge the package and all config files
sudo apt-get purge -y datadog-agent

# Remove leftover directories
sudo rm -rf /etc/datadog-agent/
sudo rm -rf /var/log/datadog/
sudo rm -rf /opt/datadog-agent/

# Remove the APT repository and key
sudo rm -f /etc/apt/sources.list.d/datadog.list
sudo rm -f /usr/share/keyrings/datadog-archive-keyring.gpg

# Refresh APT
sudo apt-get update

# Stop and remove the DCGM Exporter container
docker stop dcgm-exporter && docker rm dcgm-exporter
```

Appendix: Key File Locations

Path	Purpose
`/etc/datadog-agent/datadog.yaml`	Main agent configuration
`/etc/datadog-agent/conf.d/`	All integration check configurations
`/etc/datadog-agent/conf.d/dcgm.d/conf.yaml`	DCGM GPU check configuration
`/etc/datadog-agent/conf.d/docker.d/conf.yaml`	Docker check configuration

Path	Purpose
`/etc/datadog-agent/conf.d/nvidia.d/conf.yaml`	NVIDIA log collection configuration
`/var/log/datadog/agent.log`	Agent runtime log
`/var/log/datadog/process-agent.log`	Process agent log
`/var/log/datadog/trace-agent.log`	APM trace agent log
`/opt/datadog-agent/`	Agent binary and embedded Python runtime

Sources

- Basic Agent Usage for Ubuntu — Datadog Docs
- Supported Platforms — Datadog Docs
- Nvidia DCGM Exporter Integration — Datadog Docs
- Monitor your NVIDIA GPUs with Datadog — Datadog Blog
- GPU Monitoring Reference Architecture — Datadog
- Getting Started with the Agent — Datadog Docs
- Agent Linux Install Script — GitHub
- DGX Spark Software Stack — NVIDIA Porting Guide
- NVIDIA Container Runtime for Docker — DGX Spark User Guide
- Hardware Overview — DGX Spark User Guide