



NVIDIA Open vs Proprietary Kernel Modules: Fleet Rules

Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-10-29 · nvidia open kernel modules vs proprietary · blackwell requires open kernel modules · nvidia-open driver data center · modinfo nvidia license dual mit/gpl · nvidia open gpu kernel modules supported gpus · cuda driver compatibility

Original article: <https://gpusmith.com/articles/en/nvidia-open-gpu-kernel-modules-vs-proprietary-driver>



In this article

1. Two flavors of the same driver
 2. Architecture support by flavor and branch
 3. Features that only work with the open flavor
 4. Known costs of the open flavor
 5. Licensing, source and GSP firmware
 6. Packaging: from R560 to R615
 7. Verifying which flavor is loaded
 8. Kubernetes: the GPU Operator setting
 9. Migrating a mixed fleet
 10. Pitfalls
 11. Pre-flight checklist
 12. Sources
-

Two flavors of the same driver

The NVIDIA Linux GPU driver ships five kernel modules (`nvidia.ko`, `nvidia-modeset.ko`, `nvidia-uvm.ko`, `nvidia-drm.ko` and `nvidia-peermem.ko`), and each driver release provides them in two flavors: an **open**, source-published flavor dual licensed MIT/GPLv2, and the **proprietary** flavor NVIDIA has historically shipped ([NVIDIA driver README 615.78.08, Chapter 44](#)). The two are built from the same underlying source, are mutually exclusive in the running kernel, and should not be installed on the filesystem at the same time. User space (CUDA driver library, `nvidia-smi`, NVML) is the same with either flavor ([same README](#)).

For a fleet the flavor is a per-node-class decision: it determines whether Blackwell GPUs initialize at all, whether GPUDirect Storage and DMA-BUF RDMA work, and which packages a repository resolves. It sits one layer below the [CUDA driver/toolkit compatibility matrix](#).

Architecture support by flavor and branch

The support rules have moved twice, and the README text differs between branches. The R580 README states that "the proprietary flavor supports the GPU architectures Maxwell, Pascal, Volta, Turing, and later GPUs until Blackwell," and that Blackwell and later are supported only by the open modules ([NVIDIA driver README 580.65.06](#)). The 615.78.08 README narrows the proprietary list to "Turing, Ampere, Ada, and Hopper," again with Blackwell and later open-only ([README 615.78.08](#)). The change reflects the end of pre-Turing support: NVIDIA's data-center architecture matrix lists **R580** as the last driver branch for Maxwell, Pascal and Volta, with CUDA 12.x as their last toolkit ([NVIDIA CUDA Toolkit, Driver and Architecture Matrix, updated Sep 09, 2026](#)).

The open modules "can be used on any Turing or later GPU" ([NVIDIA/open-gpu-kernel-modules README](#)). NVIDIA's July 2024 announcement added that Grace Hopper and Blackwell platforms *must* use the open modules, that Turing, Ampere, Ada Lovelace and Hopper users are recommended to switch, and that the open modules are not compatible with Maxwell, Pascal or Volta ([NVIDIA Technical Blog, Jul 17, 2024](#)).

Table 1. Kernel module flavor by GPU architecture (as documented on 2026-10-28)

Architecture (examples)	Open flavor	Proprietary flavor	Last driver branch	Fleet rule
Maxwell, Pascal, Volta (e.g. V100)	Not supported	Required	R580	Pin to R580 proprietary
Turing, Ampere, Ada (e.g. T4, A100, L40S)	Supported, recommended	Supported (R615 README still lists it)	Ongoing	Migrate to open
Hopper (H100, H200)	Supported, recommended	Supported	Ongoing	Migrate to open
Grace Hopper (GH200)	Required	Not supported	Ongoing	Open only
Blackwell and later (B200, B300, RTX PRO 6000 Blackwell)	Required	Not supported	Ongoing	Open only

Sources: R580 and R615 READMEs, the 2024 NVIDIA blog and the architecture matrix cited above. "Last driver branch" is from the matrix; R580 is a Long Term Support Branch with end of life in June 2028, per NVIDIA's [supported drivers page](#) (revision of Sep 09, 2026).

A fleet with V100 and B200 nodes therefore cannot run one driver build everywhere: the V100 pool is capped at R580 proprietary, the B200 pool needs the open flavor.

Features that only work with the open flavor

Both the R580 and R615 READMEs list the same five features that work only with the open kernel modules ([README 615.78.08](#)):

Table 2. Open-only features and why a fleet cares

Feature	Operational consequence on proprietary modules
NVIDIA Confidential Computing	No GPU TEE mode; attestation-based deployments are blocked (see confidential GPU attestation)
Magnum IO GPUDirect Storage (GDS)	No direct NVMe/parallel-filesystem-to-GPU DMA through <code>nvidia-fs</code>
Heterogeneous Memory Management (HMM)	No transparent access to system-allocated (<code>malloc</code>) memory from GPU kernels
CPU affinity for GPU fault handlers	Fault handling cannot be pinned to chosen CPUs
DMABUF support for CUDA allocations	No DMA-BUF export of GPU memory to NICs and other devices

The right-hand column is a plain-language reading of each feature name, not NVIDIA wording. Two of these items reach well beyond the GPU itself. NVIDIA's GPU Operator documentation states that its GDS driver, version 2.17.5 or newer, "is only supported with the NVIDIA Open GPU Kernel module driver," and that the DMA-BUF path for GPUDirect RDMA requires an open kernel module driver, while the legacy `nvidia-`

peermem path works with any supported driver but requires MLNX_OFED or DOCA-OFED ([GPU Operator: GPUDirect RDMA and GPUDirect Storage](#)). NVIDIA recommends DMA-BUF over `nvidia-peermem` on that same page. A cluster that wants GDS-backed checkpointing or the recommended RDMA path is therefore on the open flavor whether or not it has Blackwell GPUs; the storage-side prerequisites are in the [GPUDirect Storage readiness matrix](#).

Known costs of the open flavor

The README is explicit that the open modules are not free of trade-offs. Compared with the proprietary modules running with GSP firmware disabled, it lists four known issues ([README 615.78.08](#)):

- GPU initialization is slower; NVIDIA suggests running `nvidia-persistenced` so GPUs are initialized before applications start.
- Entry and exit latency for S3, S4 and Runtime D3 power states can be longer, because additional GSP state is restored.
- GPU power consumption "can be marginally impacted in some scenarios."
- Turing GPUs support only the Video Memory Off flavor of Runtime D3.

On servers the first item matters most: without `nvidia-persistenced`, cold starts and health checks pay the initialization cost. On performance, the only published statement is NVIDIA's 2024 claim of "equivalent or better application performance," with no figures attached ([NVIDIA Technical Blog](#)); the switch deserves its own acceptance run.

Licensing, source and GSP firmware

The open modules are dual licensed MIT/GPLv2, and the source for every driver release is published at [NVIDIA/open-gpu-kernel-modules](#) with a tarball on [download.nvidia.com](#) ([NVIDIA Driver Installation Guide: Kernel Modules](#)). As of 2026-10-28 the GitHub README describes version 615.78.08 and lists three constraints that matter for anyone building modules themselves ([NVIDIA/open-gpu-kernel-modules](#)):

1. The modules "must be used with GSP firmware and user-space NVIDIA GPU driver components from a corresponding 615.78.08 driver release." Kernel modules and user space are version-locked to the same release.
2. The supported kernel floor is Linux 4.15 or newer.
3. The kernel interface layer must be built with the toolchain that was used to build the kernel.

The open modules are not a full open-source driver: device initialization and management are offloaded to the GPU System Processor (GSP), driven by firmware files that ship with the driver as `gsp_*.bin` under `/lib/firmware/nvidia/<version>/`. GSP firmware is used by default on all Turing and later GPUs and can be disabled only with the module parameter `NVreg_EnableGpuFirmware=0` ([README 615.78.08, Chapter 43: GSP Firmware](#)). Because the open modules require GSP, "GSP off" is a proprietary-only configuration, and image builders must ship the matching `/lib/firmware/nvidia/` tree with the modules.

Packaging: from R560 to R615

Three changes in NVIDIA's packaging explain why fleets that never chose a flavor now find themselves on one.

R560 (2024): the default flips. NVIDIA announced that from R560, installs through package managers, the runfile and the CUDA metapackage default to the open modules, with the cuda metapackage pairing the open driver from CUDA 12.6 onward ([NVIDIA Technical Blog](#)).

R570 and later: unified package names. The installation guide states that "all releases beyond driver version 570 will use these packaging conventions," with `nvidia-open` as the install target on every listed distribution (for example `dnf install nvidia-open` on RHEL 10, `apt install nvidia-open` on Debian and Ubuntu, `zypper install nvidia-open` on SUSE) and branch pinning through DNF module streams, `nvidia-driver-pinning-<branch>` on apt, or `zypper addlock` ([Driver Installation Guide: Kernel Modules, updated Sep 23, 2026](#)). The R610 edition of the guide still documents `cuda-drivers` as the meta package that "installs all NVIDIA proprietary kernel modules Driver packages," and gives per-distribution commands to switch flavors, for example `apt install --autoremove --purge cuda-drivers nvidia-driver` to go from open to proprietary on Ubuntu ([Driver Installation Guide 610: Advanced Options](#)).

R615: the proprietary packages disappear from the repository path. The R615 edition of the same page lists only `nvidia-open` in its kernel module meta-package table and has no flavor-switching section ([Driver Installation Guide 615: Advanced Options](#)). NVIDIA's CUDA apt repository for Ubuntu 24.04 shows the same change in its package metadata ([CUDA repo Packages index, ubuntu2404/x86_64](#), read 2026-10-28):

Table 3. What the proprietary-named packages resolve to (ubuntu2404 CUDA repo)

Package	610.57.04	615.71.10
<code>cuda-drivers</code>	depends on <code>nvidia-driver</code>	depends on <code>nvidia-open</code>
<code>nvidia-driver</code>	depends on <code>nvidia-dkms</code> (proprietary)	depends on <code>nvidia-driver-open</code>
<code>nvidia-dkms</code>	proprietary DKMS package	"Transitional dummy package" depending on <code>nvidia-dkms-open</code>
<code>nvidia-kernel-source</code>	proprietary kernel source	"Transitional dummy package"

On that repository, automation that installs `cuda-drivers` to get the proprietary flavor gets the open flavor once it resolves an R615 version. The runfile still offers a choice: the 615.78.08 README says installation defaults to open and that `--kernel-module-type` (short form `-M`) with `proprietary` or `open` overrides it ([README 615.78.08](#)). The README's own example shows `-m=kernel-open`, an older form that does not match its prose; scripts should use the long `--kernel-module-type=` option and check the result rather than trust either spelling. NVIDIA's RHEL 9 and SLES 15 x86_64 repositories go further: R615 ships only `kmod-nvidia-open-dkms` (DNF stream 615-open, no 615-dkms) or `nvidia-open-driver-G07`, with no proprietary kernel package and no 615 `cuda-drivers` (repository metadata read 2026-10-28).

R615 is a New Feature Branch: NVIDIA's data-center driver index lists it as "new feature branch", with 615.71.09 (2026-09-09) and 615.71.10 (2026-10-19) ([releases.json](#)). The [supported-drivers page](#) (Sep 09, 2026) does not list it yet; R580 (Long Term Support Branch, to June 2028) and R595 (Production Branch, to March 2027) are the conservative choices.

Verifying which flavor is loaded

The README gives two checks ([README 615.78.08](#)):

```
# modinfo nvidia | grep license
license:          Dual MIT/GPL          # open flavor ("NVIDIA" = proprietary)

# cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX Open Kernel Module for x86_64 [...]
```

The proprietary flavor reports NVIDIA UNIX x86_64 Kernel Module without the word "Open." modinfo reads the module file on disk for the running kernel, while /proc/driver/nvidia/version reflects the module actually loaded; after a package swap without a reboot or module reload the two can disagree, and the /proc value is the one that counts. A third check confirms GSP: nvidia-smi -q shows a GSP Firmware Version when GSP is active and "N/A" when disabled ([README 615.78.08, Chapter 43](#)).

A minimal per-node collector for the fleet inventory:

```
#!/usr/bin/env bash
# Prints: host, loaded flavor, driver version, GPU model, GSP firmware
flavor=$(grep -q 'Open Kernel Module' /proc/driver/nvidia/version && echo open || echo proprietary)
drv=$(nvidia-smi --query-gpu=driver_version --format=csv,noheader | head -1)
gpu=$(nvidia-smi --query-gpu=name --format=csv,noheader | head -1)
gsp=$(nvidia-smi -q | awk -F: ' ' '/GSP Firmware Version/{print $2; exit}')
echo "$(hostname), $flavor, $drv, $gpu, $gsp"
```

Alert on any node whose flavor differs from its node-class policy (Table 1).

Kubernetes: the GPU Operator setting

In the NVIDIA GPU Operator, the flavor is chosen with `driver.kernelModuleType`, whose valid values are `auto` (default), `proprietary` and `open`; `auto` picks the recommended type "based on the GPU devices on the host and the driver branch used." The older boolean `driver.useOpenKernelModules` is deprecated as of v25.3.0 and ignored ([GPU Operator: Getting Started](#), documentation for patch release v26.7.2). For GDS and DMA-BUF, NVIDIA advises adding `--set driver.kernelModuleType=open` when using a driver from a branch earlier than R570, and setting it explicitly to be sure the open module is used ([GPU Operator: GPUDirect RDMA and GPUDirect Storage](#)).

In a mixed cluster `auto` makes the flavor an implicit function of hardware and branch; set it explicitly per node pool and review it at each Operator upgrade.

Migrating a mixed fleet

A migration plan follows from Tables 1 to 3. An illustrative fleet makes the structure concrete (node counts are hypothetical):

Table 4. Illustrative node-class plan

Node class	Nodes	Target flavor	Branch policy	Reason
V100 training (legacy)	16	Proprietary	Pin R580	Pre-Turing; R580 is last branch
A100 inference	48	Open	Follow fleet branch	Recommended; enables DMA-BUF RDMA
H100 HGX training	64	Open	Follow fleet branch	GDS checkpointing needs open
B200 HGX	32	Open (mandatory)	Follow fleet branch	Blackwell is open-only

Of 160 nodes, 32 are already forced onto open, 16 cannot migrate, and only the 112 elective nodes (48 + 64) carry migration risk and need a canary.

A per-class sequence that respects the documented constraints:

1. **Freeze the target.** Choose the branch and record the flavor per node class. Use branch pinning (`nvidia-driver-pinning-<branch>`, DNF streams such as `nvidia-driver:580-open`, or `zypper addlock`) as documented in the [installation guide](#), so that an unpinned R615 resolution cannot change a node's flavor silently.
2. **Remove, then install.** The flavors must not coexist on disk ([README 615.78.08](#)), so use the distribution's documented switch command (`purge-and-install` on `apt`, `--allow-erasing` on `dnf`) rather than layering packages.
3. **Reboot and verify.** Confirm `/proc/driver/nvidia/version`, the GSP firmware version and the driver version with the collector above, then run the node's normal acceptance suite (NCCL, GDS and DCGM diagnostics) before returning it to the scheduler.
4. **Keep a rollback artifact.** For R610 and earlier, rollback to proprietary is a documented package switch. For R615 on repositories where proprietary names are transitional (Table 3), rollback means pinning back to an earlier branch or using the runfile with `--kernel-module-type=proprietary`; this must be decided before the upgrade, not during an incident. Pre-flight gating of the same change is described in the [CUDA production upgrade gate](#).

Pitfalls

Secure Boot and module signing. Canonical's Ubuntu documentation recommends `ubuntu-drivers` on Secure Boot systems because it installs pre-built, signed modules; it does not recommend DKMS modules unless a custom kernel requires them, because DKMS builds "are not signed with Canonical's key and thus do not support secure boot" ([Ubuntu Server docs: NVIDIA drivers installation](#)). Ubuntu ships open variants of its server drivers (for example `nvidia-driver-535-server-open`, on the same page), so the signed path is available for the open flavor too. NVIDIA's own repository installs `nvidia-dkms-open`, which builds locally and needs an enrolled signing key under Secure Boot.

Source builds. Install the `.run` user space with `--no-kernel-modules`, then build same-version source with the kernel's toolchain ([README 615.78.08](#); [NVIDIA/open-gpu-kernel-modules](#)).

Pre-Turing nodes on new branches. A V100 node that resolves a branch later than R580 has no supported driver at all ([Architecture Matrix](#)), so branch pins on legacy pools are mandatory.

Pre-flight checklist

Table 5. Flavor change pre-flight

Check	Pass condition
GPU architecture per node class	Matches Table 1; no pre-Turing GPU on an open target
Branch support	Branch is LTSB or PB on NVIDIA's supported-drivers page, or the NFB risk is accepted in writing
Package resolution	Dry-run (<code>apt-get -s</code> , <code>dnf --assumeno</code>) shows the intended <code>-open</code> or proprietary packages, and no transitional surprises
Feature needs	GDS, DMA-BUF RDMA, Confidential Computing or HMM required? Then open
Secure Boot	Signed pre-built modules available, or DKMS signing key enrolled
GSP firmware	<code>/lib/firmware/nvidia/<version>/</code> present in the image; <code>nvidia-smi -q</code> shows the GSP version
Persistence	<code>nvidia-persistenced</code> enabled
Kubernetes	<code>driver.kernelModuleType</code> set explicitly per node pool
Verification	<code>/proc/driver/nvidia/version</code> flavor matches policy on every node
Rollback	Pinned previous branch or runfile path tested on one canary node

GPU Smith can review a fleet's driver and kernel-module plan against these documented constraints as part of an upgrade engagement.

Sources

1. [Chapter 44. Open Linux Kernel Modules, NVIDIA driver README 615.78.08](#) - NVIDIA; flavors, R615 architecture support, open-only features, known issues, installer options, verification (accessed 2026-10-28)
2. [Chapter 43. GSP Firmware, NVIDIA driver README 615.78.08](#) - NVIDIA; GSP default on Turing+, firmware path, `NVreg_EnableGpuFirmware`, `nvidia-smi -q` check (accessed 2026-10-28)
3. [Open Linux Kernel Modules, NVIDIA driver README 580.65.06](#) - NVIDIA; R580 architecture support wording (accessed 2026-10-28)
4. [NVIDIA/open-gpu-kernel-modules \(GitHub README\)](#) - NVIDIA; version 615.78.08, Turing+ support, GSP and user-space version lock, kernel 4.15+, toolchain rule (accessed 2026-10-28)

5. [NVIDIA Transitions Fully Towards Open-Source GPU Kernel Modules](#) - NVIDIA Technical Blog, Jul 17, 2024; R560 default, Grace Hopper/Blackwell requirement, pre-Turing incompatibility, performance claim (accessed 2026-10-28)
6. [CUDA Toolkit, Driver and Architecture Matrix](#) - NVIDIA, updated Sep 09, 2026; R580 last branch for Maxwell, Pascal, Volta (accessed 2026-10-28)
7. [Supported Drivers and CUDA Toolkit Versions](#) - NVIDIA, updated Sep 09, 2026; branch designations and end-of-life dates (accessed 2026-10-28); release index with R615 branch type:
<https://docs.nvidia.com/datacenter/tesla/drivers/releases.json>
8. [Driver Installation Guide: Kernel Modules](#) - NVIDIA, updated Sep 23, 2026; MIT/GPLv2 licensing, post-570 packaging, install and pinning commands (accessed 2026-10-28)
9. [Driver Installation Guide 610: Advanced Options](#) - NVIDIA; cuda-drivers proprietary meta package, flavor-switching commands (accessed 2026-10-28)
10. [Driver Installation Guide 615: Advanced Options](#) - NVIDIA; R615 meta-package table listing only nvidia-open (accessed 2026-10-28)
11. [CUDA repository Packages index, ubuntu2404/x86_64](#) - NVIDIA; package dependencies for 610.57.04 and 615.71.10 (read 2026-10-28); RHEL 9 and SLES 15 repodata:
https://developer.download.nvidia.com/compute/cuda/repos/rhel9/x86_64/repodata/repomd.xml and
https://developer.download.nvidia.com/compute/cuda/repos/sles15/x86_64/repodata/repomd.xml
12. [GPU Operator: Getting Started](#) - NVIDIA, v26.7.2 docs; driver.kernelModuleType values and default, useOpenKernelModules deprecation (accessed 2026-10-28)
13. [GPU Operator: GPUDirect RDMA and GPUDirect Storage](#) - NVIDIA; GDS 2.17.5+ open-only, DMA-BUF prerequisites, pre-R570 guidance (accessed 2026-10-28)
14. [NVIDIA drivers installation, Ubuntu Server documentation](#) - Canonical; signed pre-built modules, DKMS and Secure Boot, -server-open packages (accessed 2026-10-28)

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://download.nvidia.com/XFree86/Linux-x86_64/615.78.08/README/kernel_open.html
2. https://download.nvidia.com/XFree86/Linux-x86_64/580.65.06/README/kernel_open.html
3. <https://docs.nvidia.com/datacenter/tesla/drivers/cuda-toolkit-driver-and-architecture-matrix.html>
4. <https://github.com/NVIDIA/open-gpu-kernel-modules>
5. <https://developer.nvidia.com/blog/nvidia-transitions-fully-towards-open-source-gpu-kernel-modules>
6. <https://docs.nvidia.com/datacenter/tesla/drivers/supported-drivers-and-cuda-toolkit-versions.html>
7. <https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-operator-rdma.html>
8. <https://docs.nvidia.com/datacenter/tesla/driver-installation-guide/kernel-modules.html>
9. https://download.nvidia.com/XFree86/Linux-x86_64/615.78.08/README/gsp.html
10. <https://docs.nvidia.com/datacenter/tesla/driver-installation-guide/610/advanced-options.html>
11. <https://docs.nvidia.com/datacenter/tesla/driver-installation-guide/615/advanced-options.html>
12. https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/x86_64/Packages.gz

13. <https://docs.nvidia.com/datacenter/tesla/drivers/releases.json>
14. <https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/getting-started.html>
15. <https://ubuntu.com/server/docs/how-to/graphics/install-nvidia-drivers>
16. https://developer.download.nvidia.com/compute/cuda/repos/rhel9/x86_64/repodata/repomd.xml
17. https://developer.download.nvidia.com/compute/cuda/repos/sles15/x86_64/repodata/repomd.xml

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the [hardware library](#), [GPU references](#), [networking references](#) and [vendor research](#). [Contact GPU Smith](#) with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: <https://gpusmith.com>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.