

NEXAVM CLOUD | OPERATIONS MANUAL

NVIDIA pGPU and vGPU Operations Manual

A controlled procedure for deploying, attaching, validating, and maintaining NVIDIA physical GPU passthrough and NVIDIA virtual GPU workloads on NexaVM Cloud.

VERSION

1.6

PLATFORM

NexaVM Cloud

AUDIENCE

Cloud and GPU Platform Operators

CLASSIFICATION

Operational Guidance

1 Purpose and allocation choice

Use this manual for planned implementation. It combines the NexaVM Cloud operational flow with NVIDIA KVM prerequisites and current NexaVM and NVIDIA documentation links. Complete the relevant pre-change checks before changing host firmware, drivers, or allocations.

Non-negotiable allocation rule. NexaVM Cloud VM must not have both a pGPU and a vGPU attached. A NexaVM VM can attach one vGPU at a time. Use a stopped VM for vGPU attach and detach; stop a VM before pGPU detach as the operating standard.

Allocation model	Use when	Operational characteristics
pGPU passthrough	A VM needs exclusive ownership, direct device behavior, or a workload is not suitable for vGPU.	One VM can attach multiple pGPUs. Mobility and HA behavior may be limited after passthrough.
NVIDIA vGPU	Multiple workloads must share a compatible physical GPU using NVIDIA-defined profiles.	The physical GPU is virtualized first, then a vGPU profile is allocated to a VM. NVIDIA licensing and host/guest driver compatibility are mandatory controls.

2 Pre-change checks

Compatibility control. An incompatible NVIDIA guest driver can prevent a vGPU from loading. Do not treat a new driver as automatically compatible.

Step	Area	Requirement
1	Firmware	Enable virtualization support and IOMMU.
2	Host readiness	Confirm kernel, GPU visibility, and driver prerequisites.
3	NexaVM readiness	Enable IOMMU and confirm status is Available.
4	Allocation	Use either pGPU or vGPU workflow, never both.

Record GPU model, PCI address, server model, NexaVM host kernel, guest OS, NexaVM Cloud version, and target workload. This manual treats the NexaVM host as RHEL-family and uses `yum`, `dracut`, and `systemd` commands accordingly.

- ☐ Choose the allocation model before downloading any driver. pGPU and vGPU use different host-side software and different driver-selection paths.
- ☐ Confirm GPU, platform, hypervisor, and guest OS support in the NVIDIA vGPU Product Support Matrix.

- ❑ For vGPU, obtain the required NVIDIA vGPU software entitlement and licenses before deployment. Confirm access to the NVIDIA Licensing Portal or Enterprise Application Hub, the correct software release, sufficient license capacity, and the planned NVIDIA License System service.
- ❑ Schedule host reboot and VM outage. Preserve remote console access before any driver change.
- ❑ Capture current NexaVM allocation state and `nvidia-smi` output.

3 Common host and platform preparation

3.1 Firmware requirements

- Enable **Intel VT-d** or **AMD IOMMU** in BIOS or UEFI.
- Enable **Above 4G Decoding**, or the server-equivalent MMIO-above-4-GB option, when required for the installed GPU or its large BAR address space.
- Do not infer the NVIDIA package version from a GPU family name alone. Record the exact PCI device identity, server model, and running NexaVM host kernel before choosing the driver release.

3.2 NexaVM UI preparation

Task	NexaVM path	Required result
New host	Resource Center > Hardware > Computing Facility > Host > Add Host	Set Scan Host IOMMU Setting to true.
Existing host	Resource Center > Hardware > Computing Facility > Host > select host	Record the current state, set IOMMU State to true, reboot the host, reconnect it, and verify IOMMU Status is Available.

1. Open the host **Overview** page and expand **Basic Info**. Record **IOMMU State** and **IOMMU Status** before making the change.
2. If the state is disabled or the status is unavailable, first verify VT-d or AMD IOMMU in server firmware. Then enable the NexaVM **IOMMU State** switch.
3. Place or keep the host in maintenance as required by the change plan, evacuate or stop affected VMs, and reboot the host.
4. After the host reconnects, return to the same page. Continue only when **IOMMU State** is enabled and **IOMMU Status** is Available.

Figure 1. Locate IOMMU State and IOMMU Status on the host Overview page.

The original manual's supplied image for this figure (NexaVM host Overview page, Basic Info) did not render in the source PDF. It showed the pre-configuration condition: the switch is disabled and status is Unavailable. That is not a successful end state. Enable the setting, reboot the host, and verify Available before assigning a pGPU or creating vGPUs.

```
uname -r
lspci -nn | grep -Ei "vga|3d|nvidia"
dmesg | grep -Ei "iommu|dmar|amd-vi"
```

Do not continue when IOMMU State is enabled but IOMMU Status is unavailable. If the status remains unavailable after the planned reboot, recheck firmware virtualization settings, the host boot parameters and kernel log, and the platform support combination before allocation.

3.3 Choose the driver's path before installation

Deployment	KVM host or NexaVM host	Guest VM	Where to obtain the accurate package
pGPU passthrough	NexaVM uses the platform VFIO or vfio-pci path to assign the GPU. The NexaVM tutorial additionally requires the supported host GPU driver and monitoring integration when real-time GPU load monitoring is required. Do not install Virtual GPU Manager solely for pGPU passthrough.	Install the standard NVIDIA driver that supports the passed-through GPU, guest OS, and workload. Install the supported NexaVM GuestTools components required for guest monitoring.	Use the NexaVM GPU Passthrough Tutorial together with NVIDIA Driver Downloads or the enterprise driver repository. Verify the exact GPU, host/guest OS, monitoring requirement, and package role before installation.
NVIDIA vGPU	The NexaVM tutorial distinguishes the NVIDIA host GPU driver and the NVIDIA vGPU Manager or KVM vGPU driver. Obtain and install the supported components for the approved NexaVM host, kernel, GPU, and vGPU release.	Install the NVIDIA vGPU guest driver from the same vGPU release bundle or another combination explicitly allowed by that release's compatibility policy. Do not install Virtual GPU Manager in the VM.	Start with the NexaVM vGPU Virtualization Tutorial, NVIDIA vGPU Product Support Matrix, and KVM release notes. Download the required host GPU driver, KVM vGPU Manager, and VM guest driver from the authorized NVIDIA source.

Stop if the package role is unclear. A filename containing `-vgpu-kvm` identifies the host-side KVM vGPU component, not a VM guest driver. The NexaVM tutorial may list a separate NVIDIA host GPU driver for monitoring and vGPU preparation. Record the purpose, install location, download page, release branch, filename, and checksum of every package.

4 Complete pGPU passthrough workflow

Follow this section from start to finish for exclusive physical GPU assignment. Do not perform the vGPU Manager installation or NexaVM Virtualization action in Section 5.

4.1 Confirm the pGPU driver plan

1. On the host, record the exact GPU name, PCI address, vendor and device ID with `lspci -nn | grep -Ei "vga|3d|nvidia"`. Record the target guest OS, version, architecture, and workload.
2. Confirm that NexaVM Cloud, the server, IOMMU configuration, GPU, guest firmware, and guest OS support passthrough. For GPUs with a large BAR, use UEFI guest firmware and confirm the required MMIO or Above 4G configuration.

3. From NVIDIA Driver Downloads or the approved enterprise repository, filter by the exact GPU product and guest OS. Open the driver details and verify that the GPU appears in Supported Products and that the guest OS and architecture match.
4. Record the selected standard guest-driver version, filename, download source, and checksum. Do not select a `-vgpu-kvm` package; that package is Virtual GPU Manager for a vGPU host.

4.2 Prepare NexaVM for pGPU

1. Open **Settings > Platform Settings > Global Settings**. Apply **Hide KVM Virtualization Flag** and **VM Hyper V Virtualization** according to the validated NVIDIA pGPU design and guest requirements.
2. Review PCI hot plugging policy. Use cold attach and cold detach unless the complete GPU, guest OS, driver, and workload combination has been validated for hot plug.
3. Confirm the target GPU is not allocated to another VM and that NexaVM reports it as available. If host-side GPU monitoring is required, install the host GPU driver and monitoring components specified by the applicable NexaVM tutorial, then verify that NexaVM can still prepare and bind the selected device for passthrough.

4.3 Detect and optionally configure the physical GPU

1. Open **Resource Center > Hardware > Computing Facility > Host** and select the prepared host.
2. Open **Associated Resource > PCI Device > Physical GPU**. Confirm the pGPU is detected and available.
3. Open **Resource Center > Resource Pool > Compute Configuration > GPU Specification**. Confirm the scanned pGPU specification matches the hardware.
4. Use **Actions > Set ROM** only when the vendor or validated design requires it. The file must match the exact GPU specification and version.

4.4 Attach the pGPU to a stopped VM

1. For a new VM, open **Resource Center > Resource Pool > Virtual Resource > VM Instance > Create VM Instance**.
2. Under **Resource Configurations**, select **Attach GPU Specification** or **Attach GPU Device**. Confirm that the complete GPU device group required by the hardware—including associated graphics, audio, USB, or other functions—is handled as required by the NexaVM inventory and validated design.
3. For an existing VM, stop the VM, then open **VM details > Configuration Info > pGPU Device > Attach**.
4. Confirm the selected PCI identity matches the change record and that no vGPU is attached. Start the VM only after the allocation is correct.

4.5 Install the standard NVIDIA driver inside the pGPU VM

Linux guest. Install the compiler and the kernel development or header packages that exactly match the running guest kernel. Disable Nouveau only if it is loaded and the selected NVIDIA driver requires it. For RHEL-family guests rebuild initramfs with `dracut --force`; for Debian or Ubuntu use `update-initramfs -u`.

```
# Run inside the pGPU guest, not on the NexaVM host
uname -r
lspci -nn | grep -i nvidia
lsmod | grep nouveau
# Install the exact prerequisites for this guest OS and kernel first
chmod +x NVIDIA-Linux-x86_64-<approved-standard-driver>.run
./NVIDIA-Linux-x86_64-<approved-standard-driver>.run
```

```
reboot
nvidia-smi
```

Windows guest. Use UEFI firmware when required by the device or large BAR configuration. Install the approved standard NVIDIA Windows driver that lists the exact GPU and Windows release, reboot, and confirm the device has no error in Device Manager before running `nvidia-smi`.

4.6 Validate pGPU and hand over

1. In NexaVM, verify that the VM shows the intended pGPU and no vGPU.
2. Inside the guest, run `nvidia-smi` and confirm the expected GPU model, driver version, and usable framebuffer.
3. Run a representative workload and check the guest and host logs for PCI, VFIO, Xid, or driver-reset errors.
4. Save the PCI identity, allocation screenshot, guest-driver package and version, `nvidia-smi` output, and workload result.

Detach control. Detaching a pGPU from a running VM can cause a guest blue screen or suspension. Stop the VM before detach. Review host change, host and primary storage change, and HA implications before production assignment.

5 Complete NVIDIA vGPU workflow

Follow this section from start to finish when a supported physical GPU will provide NVIDIA-defined vGPU profiles. Do not attach that physical GPU directly to the VM as a pGPU.

5.1 Select the supported vGPU release and all required drivers

1. Record the exact GPU model and PCI identity, NexaVM host kernel, server model, guest OS, and required workload profile.
2. Open the NVIDIA vGPU Product Support Matrix for the intended release family. Confirm the GPU, Linux with KVM support, guest OS, and required vGPU profile are supported together. Also check the GPU support-lifecycle page and release notes for known issues.
3. In the NVIDIA Licensing Portal or Enterprise Application Hub, choose the vGPU software release validated in the previous step. Following the NexaVM tutorial and NVIDIA release packaging, obtain the required NVIDIA host GPU driver, Linux KVM Virtual GPU Manager or vGPU KVM driver, and Linux or Windows vGPU guest driver.
4. Prefer host and guest components from the same release. If versions differ, verify the exact pair against the release's cross-branch compatibility table. Record every component version, filename, source page, purpose, and checksum.

Package	Install location	Identification check
NVIDIA host GPU driver	NexaVM KVM host	The NexaVM vGPU tutorial lists a host GPU driver in addition to the KVM vGPU component. Select the version validated for the NexaVM host, kernel, GPU, monitoring requirement, and chosen vGPU release; do not assume an example version is universal.

Package	Install location	Identification check
Virtual GPU Manager or KVM vGPU driver	NexaVM KVM host	Package is explicitly for Linux with KVM and the approved host distribution. A generic KVM runfile commonly includes -vgpu-kvm; supported RHEL deployments may provide an RPM.
vGPU guest driver	VM after a vGPU is attached	Package is the Linux or Windows guest driver from the approved vGPU software release, and its release is compatible with the installed Manager.

5.2 Prepare the host firmware and NexaVM vGPU settings

1. In addition to the common IOMMU preparation, enable SR-IOV only when the exact GPU and NVIDIA deployment guide require it. Enable Above 4G Decoding or the server-equivalent MMIO option when required.
2. Apply model- and release-specific host prerequisites conditionally. For example, the linked NexaVM vGPU tutorial specifies host kernel 4.18 and GCC 8.3.1 for NVIDIA A-series GPUs. Verify that requirement against the deployed NexaVM release and exact GPU instead of applying it to unrelated models.
3. Open **Settings > Platform Settings > Global Settings**. Configure the approved vGPU Default Quota and review PCI hot plugging policy.
4. Confirm the physical GPU is available and not attached to a VM as pGPU. Remove conflicting allocations before installing or generating vGPU devices.

5.3 Install and verify the NVIDIA host components

Install the NVIDIA host GPU driver and the Virtual GPU Manager or KVM vGPU driver required by the NexaVM tutorial and the selected NVIDIA release. These are host components; neither is the VM guest driver. The filenames below show package roles only and are not version recommendations.

5.3.1 Find and record the approved version

1. Identify the exact GPU model and PCI ID, NexaVM host distribution and kernel, guest OS, and required vGPU profile.
2. Use the linked NexaVM vGPU tutorial, NVIDIA vGPU Product Support Matrix, and the Linux with KVM release notes to select a release that supports the entire combination.
3. From the NVIDIA Licensing Portal, Enterprise Application Hub, or an approved NexaVM repository, copy the exact KVM package URL. Record the release as `<approved-version>`, for example 535.183.04 only when that exact version has passed the compatibility check.
4. Confirm that the KVM package name follows the expected role, such as `NVIDIA-Linux-x86_64-<approved-version>-vgpu-kvm.run`. Do not infer compatibility from the filename alone.

5.3.2 Standard NexaVM host installation procedure

NexaVM hosts use the RHEL-family package and boot-management workflow shown below. Run it from a root shell during an approved maintenance window with out-of-band console access. Keep the procedure unchanged unless NexaVM support directs otherwise. Replace only the NVIDIA version and package URL with values validated for the installed GPU and chosen vGPU release.

```
# 1. Record the environment and set the version selected from the matrices
uname -r
```

```
lspci -nn | grep -Ei "vga|3d|nvidia"
systemctl get-default
VGPU_VERSION="approved-version"
VGPU_PACKAGE="NVIDIA-Linux-x86_64-${VGPU_VERSION}-vgpu-kvm.run"
VGPU_URL="approved-package-url"
# Example only after validation:
# VGPU_VERSION="535.183.04"
# VGPU_PACKAGE="NVIDIA-Linux-x86_64-${VGPU_VERSION}-vgpu-kvm.run"

# 2. Disable Nouveau without appending duplicate configuration lines
cat > /etc/modprobe.d/nvidia-installer-disable-nouveau.conf <<'EOF'
blacklist nouveau
options nouveau modeset=0
EOF

# 3. Install packages matching the running host kernel
yum install -y gcc make elfutils-libelf-devel \
    kernel-devel-$(uname -r) kernel-headers-$(uname -r)
# Confirm that the matching kernel packages are present before continuing
rpm -q kernel-devel-$(uname -r) kernel-headers-$(uname -r)

# 4. Back up and rebuild the current initramfs
cp -a /boot/initramfs-$(uname -r).img \
    /boot/initramfs-$(uname -r).img.bak
dracut /boot/initramfs-$(uname -r).img $(uname -r) --force

# 5. Boot into non-graphical mode for driver installation
systemctl set-default multi-user.target
systemctl isolate multi-user.target
# On older systems, "init 3" is the equivalent transition.

# 6. Download the exact package selected above
wget -O "${VGPU_PACKAGE}" "${VGPU_URL}"
test -s "${VGPU_PACKAGE}"
# Verify the vendor-published checksum here when one is available.

# 7. Install the KVM vGPU package; accept defaults only if they match
# the approved change plan and the installer reports no conflict.
chmod +x "${VGPU_PACKAGE}"
bash "${VGPU_PACKAGE}"

# 8. Reboot and validate the host
reboot
# Run after the host returns
lsmod | grep nouveau          # Expected: no output
nvidia-smi
systemctl status nvidia-vgpu --no-pager
```

If the host previously used a graphical default target, restore it only after validation with `systemctl set-default graphical.target`. Do not continue if Nouveau remains loaded, the installer reports a kernel mismatch, `nvidia-smi` does not report the physical GPU, or the service and modules required by the selected release are unhealthy.

Install the compiler, kernel development package, and kernel headers required by the selected release and matching the running host kernel. If Nouveau is loaded, disable it and rebuild initramfs using the host distribution's documented procedure. Install the approved NVIDIA host GPU driver when it is a separate package in the validated NexaVM/NVIDIA stack. Then install the approved KVM Manager package, or use the release-specific packages and order required by the supported platform documentation. Reboot when required. Run

`nvidia-smi`, confirm `nvidia-vgpu` is active when used by the release, and verify the NVIDIA vGPU modules required by the release are loaded.

5.4 Virtualize the pGPU in NexaVM

Options in the Virtualization dialog depend on the discovered GPU, installed vGPU Manager, NVIDIA release, and available NexaVM GPU specifications. A GPU model name by itself does not determine the correct choice. Select only an option whose device model, profile name, framebuffer, maximum instances, display capability, and release support match the approved workload design.

Option to check	How to determine the correct value	Reject the choice when
Physical GPU	Match the NexaVM inventory to the host PCI address and <code>nvidia-smi</code> device identity.	The device is allocated, not virtualizable, or its identity differs from the change record.
GPU or vGPU specification	Match the exact NVIDIA profile supported for the installed GPU and vGPU software release; verify framebuffer and intended workload.	The profile is absent from the support matrix, belongs to another GPU, or its framebuffer/capacity is unsuitable.
Instance count	Use the profile capacity reported for that physical GPU and reserve capacity for the approved placement plan.	The requested count exceeds available capacity or conflicts with existing allocations.
Display capability	Choose a display-enabled or compute-oriented profile according to the workload and NVIDIA profile definition.	The selected capability does not match the application or guest requirements.

1. On the prepared host, open **Associated Resource > PCI Device > Physical GPU**.
2. Select an unallocated, virtualizable NVIDIA pGPU. Choose **Actions > Virtualization**.
3. Compare every offered specification with the approved support-matrix entry and workload record. Confirm the exact profile name, instance count, framebuffer, and display capability; do not copy the selection shown in an example screenshot.
4. Wait for completion. Open **Associated Resource > PCI Device > vGPU** and verify the generated devices.
5. Open **GPU Specification > vGPU Specification** and verify allocation availability.
6. If the vGPU devices or specifications are still not visible after the operation completes, return to the host page and choose **Actions > Reconnect**. Wait for the host to reconnect and rescan, then check the vGPU inventory again.

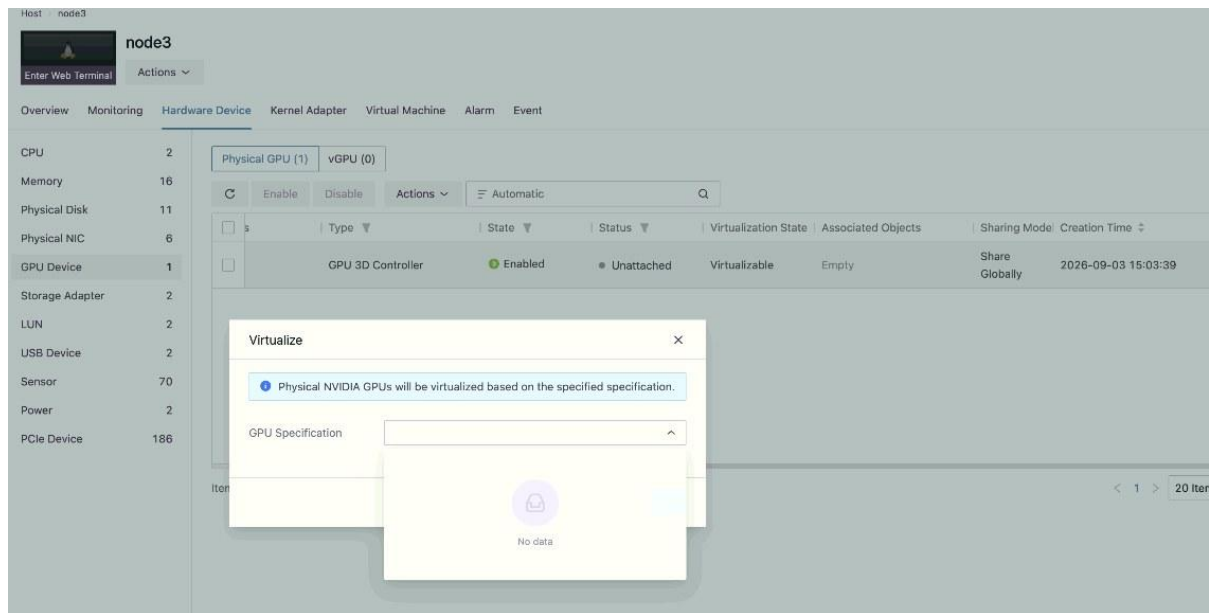


Figure 2. Physical GPU before vGPU configuration is complete. This image shows the condition in which vGPU configuration has not been completed and the physical GPU is unavailable for slicing. Do not treat this as a successful virtualization result. Complete the supported vGPU Manager and host configuration, reconnect the host in NexaVM if necessary, and confirm the GPU becomes virtualizable before selecting a validated vGPU specification.

5.5 Attach one vGPU to a stopped VM

1. Stop the VM. vGPU attach and detach require the Stopped state.
2. For a new VM, choose **Create VM Instance > Resource Configurations**. For an existing VM, choose **VM details > Configuration Info > vGPU Device > Attach**.
3. Select the approved vGPU specification or exact device. Confirm the VM has no pGPU and no other vGPU attached.
4. Start the VM only after the allocation is correct. Continue directly to Section 5.6.

5.6 Install the compatible vGPU driver inside the VM

Linux guest. Install the guest-kernel prerequisites required by the selected driver. Disable Nouveau only if it is loaded and the NVIDIA instructions require it. The package below is the vGPU guest driver selected in Section 5.1, not the host `-vgpu-kvm` package.

```
# Run inside the vGPU VM, not on the NexaVM host
uname -r
lspci -nn | grep -i nvidia
lsmod | grep nouveau
chmod +x NVIDIA-Linux-x86_64-<approved-vgpu-guest-driver>.run
./NVIDIA-Linux-x86_64-<approved-vgpu-guest-driver>.run
reboot
nvidia-smi
```

Windows guest. Install the Windows vGPU guest driver from the approved vGPU release bundle, reboot, and confirm the vGPU appears without a Device Manager error. Do not substitute a standard consumer or data-center package unless the release documentation explicitly identifies it as the supported vGPU guest package.

5.7 Configure and verify mandatory NVIDIA vGPU licensing

A valid NVIDIA vGPU software entitlement and sufficient vGPU license capacity are required for this workflow. Deploy or identify the NVIDIA License System service, allocate the correct license type, configure the guest

license client, verify network, DNS, and time synchronization, and confirm license acquisition in the guest before service acceptance. Do not hand over an unlicensed or reduced-capability vGPU VM.

5.8 Validate vGPU and hand over

1. On the host, save `nvidia-smi`, Manager version, service state, and module evidence.
2. In NexaVM, verify the generated vGPU devices and profile capacity, and confirm the VM has one intended vGPU and no pGPU.
3. Inside the guest, run `nvidia-smi` and confirm the expected vGPU profile, compatible guest-driver version, framebuffer, and active license state.
4. Run a representative workload and save the host, NexaVM, guest, licensing, and workload evidence.

5.9 Supplied host evidence

```
[root@node3 ~]# nvidia-smi
Tue Sep  8 13:18:31 2026

+-----+
| NVIDIA-SMI 595.91.04                  Driver Version: 595.91.04      CUDA Version: N/A        |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                               Persistence-M   Bus-Id       Disp.A     Volatile Uncorr. ECC  |
| Fan  Temp  Perf          Pwr:Usage/Cap     |      00000000:81:00:0  Off         0%          Default  |
|-----+-----+-----+-----+-----+-----+
|  0  NVIDIA A40                               On          |      00000000:81:00:0  Off         0%          Default  |
|      36C    P8              31W / 300W     |                      |                      |
+-----+-----+-----+-----+-----+-----+

Processes:
+-----+-----+-----+-----+-----+-----+
| GPU   GI      CI           PID    Type    Process name                      |
| ID    ID                               |                      |                      |
+-----+-----+-----+-----+-----+-----+
|      0   0   0               0      Type    Process name                      |
|      0   0   0               0      Type    Process name                      |
+-----+-----+-----+-----+-----+-----+
| No running processes found |
+-----+-----+-----+-----+-----+-----+

[root@node3 ~]#
```

Figure 3. Host driver verification example. `nvidia-smi` confirms the host sees the physical GPU. Record the detected model, driver version, and vGPU software release in the change record.

```
[root@node3 ~]# lsmod | grep nvidia
nvidia_vgpu_vfio 98384 12
nvidia 98807808 3
ndev 24576 2 vfio_mdev,nvidia_vgpu_vfio
vfio 36864 4 vfio_mdev,nvidia_vgpu_vfio,vfio_iommu_type1,vfio_pci
irqbypass 16384 3 nvidia_vgpu_vfio,vfio_pci,kvm
drm 602112 11 gpu_sched,drm_kms_helper,ast,drm_shmem_helper,drm_display_helper,nvidia,drm_buddy,amdgpu,drm_ttm_helper,ttm
[root@node3 ~]# ls /sys/bus/pci/devices/0000:81:00.4/mdev_supported_types/
ls: impossible accedere a '/sys/bus/pci/devices/0000:81:00.4/mdev_supported_types/': No such file or directory
[root@node3 ~]# systemctl status nvidia-vgpusd | grep nvidia
Unit nvidia-vgpusd.service could not be found.
[root@node3 ~]# ls /sys/bus/pci/devices/0000:81:00.4/mdev_supported_types/
ls: impossible accedere a '/sys/bus/pci/devices/0000:81:00.4/mdev_supported_types/': No such file or directory
[root@node3 ~]# systemctl status nvidia-vgpusd
● nvidia-vgpusd.service - NVIDIA vGPU Daemon
   Loaded: loaded (/usr/lib/systemd/system/nvidia-vgpusd.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Tue 2026-09-08 13:08:14 CEST; 33min ago
   Main PID: 2769 (code=exited, status=0/SUCCESS)

set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: Encoder Capacity: 0x64
set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: BAR1 Length: 0x100
set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: Frame Rate Limiter enabled: 0x1
set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: Number of Displays: 4
set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: Max pixels: 36864000
set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: Display: width 5120, height 2880
set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: Multi-vGPU Exclusive supported: 0x1
set 08 13:08:14 node3.c3usterfuck nvidia-vgpusd[2769]: License: GRID-Virtual-WS,5.0;GRID-Virtual-WS,2.0;GRID-Virtual-WS-Ext,2.0
set 08 13:08:14 node3.c3usterfuck systemd[1]: nvidia-vgpusd.service: Succeeded.
set 08 13:08:14 node3.c3usterfuck systemd[1]: Started NVIDIA vGPU Daemon.
[root@node3 ~]# cat > /usr/systemd/system/nvidia-vgpusd.service.c[1]EOF
```

Figure 4. vGPU host service and module evidence example. NVIDIA vGPU VFIO modules and `nvidia-vgpusd` are active. This complements NexaVM and guest validation.

Restore to pGPU control. Detach every related vGPU from all VM instances before choosing **Actions > Ungenerate**.

6 Final validation and customer acceptance

Check	pGPU acceptance result	vGPU acceptance result
Host readiness	IOMMU Status is Available; the pGPU appears in NexaVM inventory and can be prepared for the VFIO passthrough path. When NexaVM GPU load monitoring is required, verify the supported host GPU driver and monitoring integration before assignment. Guest <code>nvidia-smi</code> remains the definitive driver check after exclusive assignment.	IOMMU Status is Available; host <code>nvidia-smi</code> reports GPU and approved driver.
NexaVM allocation	VM shows intended pGPU and no vGPU.	vGPU devices and specifications are visible; VM shows one vGPU and no pGPU.
Guest driver	Guest <code>nvidia-smi</code> reports GPU and correct driver.	Guest <code>nvidia-smi</code> reports vGPU and compatible driver.
Licensing	Apply when selected pGPU use requires it.	Guest has acquired NVIDIA license with no reduced-capability state.
Workload	Representative workload completes without driver reset or PCI error.	Representative workload completes; expected monitoring is available.

Useful evidence commands: `nvidia-smi`, `nvidia-smi -q`, and `lspci -nn | grep -i nvidia`.

- ☐ For pGPU, save host PCI and VFIO evidence plus guest `nvidia-smi`. For vGPU, save host and guest `nvidia-smi` plus Manager service evidence.
- ☐ Save license-client evidence for each licensed vGPU VM.
- ☐ Confirm GPU monitoring after host, guest, and GuestTools installation.
- ☐ Capture successful representative workload output before handover.

7 Troubleshooting and controlled recovery

Symptom	Likely cause	Corrective action
IOMMU Status unavailable	Host was not rebooted or firmware setting is ineffective.	Reboot. If unresolved, correct firmware settings and validate IOMMU kernel logs.
Nouveau in use	Nouveau is loaded or initramfs was not rebuilt.	Blacklist Nouveau, rebuild initramfs, reboot, and confirm no <code>lsmod</code> output.
Kernel source tree error	Headers or development package does not match kernel.	Install exact matching packages or boot a compatible kernel.
vGPU does not initialize	Host Manager and guest driver incompatible or profile unsupported.	Return to NVIDIA compatibility matrix and use approved pair and profile.

Symptom	Likely cause	Corrective action
vGPU or vGPU specification is not visible in NexaVM	The host inventory has not refreshed after vGPU Manager installation or GPU virtualization, or host configuration is incomplete.	First verify the Manager, required service and modules, and GPU virtualization result. Then open the host in NexaVM, choose Actions > Reconnect , wait for the rescan to finish, and check the vGPU inventory again.
vGPU degraded	License, client token, network, DNS, or time problem.	Validate entitlement, guest configuration, connectivity, and time synchronization.
GPU attach fails	VM is running, allocation conflicts, quota, or capacity issue.	Stop VM; remove conflict; check quota and inventory; retry with valid device or profile.

Controlled rollback

1. Stop affected VMs and preserve host/guest `nvidia-smi`, installer logs, NexaVM events, and license evidence.
2. For vGPU rollback, detach all vGPUs before Ungenerate. Restore the last approved host package and verify host `nvidia-smi`.
3. Re-create profiles only after compatibility and license prechecks pass.

NVIDIA virtualization references

With the current NVIDIA documentation for the exact hardware and software release selected for the implementation.

- [NVIDIA Driver Downloads](#) – standard guest-driver selection for pGPU passthrough.
- [NVIDIA vGPU software releases](#) – current release branches and component versions.
- [NVIDIA GPUs supported by vGPU](#) – GPU support lifecycle and applicable release families.
- [NVIDIA vGPU Deployment Guides](#) – entry point for current KVM deployment guides.
- [NVIDIA vGPU Deployment Guide for Red Hat Enterprise Linux with KVM](#) – RHEL KVM configuration, licensing, and validation.
- [NVIDIA vGPU Deployment Guide for Ubuntu with KVM](#) – Ubuntu KVM prerequisites and deployment flow.
- [Using GPU Pass Through on a Linux with KVM Hypervisor](#) – NVIDIA KVM passthrough guidance.
- [NVIDIA Virtual GPU Software User Guide](#) – Manager installation, guest drivers, mediated-device, and SR-IOV procedures.
- [NVIDIA vGPU Product Support Matrix](#) – hardware, hypervisor, and guest support verification.
- [NVIDIA Linux with KVM Release Notes](#) – compatibility requirements and release-specific issues.
- [NVIDIA Virtual GPU Software Documentation](#) – licensing, user guide, and management documentation index.

Source baseline: NexaVM Cloud GPU Passthrough and vGPU Passthrough tutorials. Current NVIDIA references are linked in this guide. Supplied screenshots are included as configuration and validation examples.