Red Hat Virtualization 4.3

Setting up an NVIDIA GPU for a virtual machine in Red Hat Virtualization

How to configure a virtual machine in Red Hat Virtualization to use a dedicated GPU or vGPU.

Last Updated: 2022-10-30

Red Hat Virtualization 4.3 Setting up an NVIDIA GPU for a virtual machine in Red Hat Virtualization

How to configure a virtual machine in Red Hat Virtualization to use a dedicated GPU or vGPU.

Red Hat Virtualization Documentation Team
Red Hat Customer Content Services
rhev-docs@redhat.com

Legal Notice

Abstract

This document describes how to use a host with a graphics processing unit (GPU) to run virtual machines in Red Hat Virtualization for graphics-intensive tasks and software that cannot run without a GPU.

Table of Contents

PREFACE

You can use a host with a compatible graphics processing unit (GPU) to run virtual machines in Red Hat Virtualization that are suited for graphics-intensive tasks and for running software that cannot run without a GPU, such as CAD.

You can assign a GPU to a virtual machine in one of the following ways:

- **GPU passthrough**: You can assign a host GPU to a single virtual machine, so the virtual machine, instead of the host, uses the GPU.
- **Virtual GPU (vGPU)**: You can divide a physical GPU device into one or more virtual devices, referred to as *mediated devices*. You can then assign these mediated devices to one or more virtual machines as virtual GPUs. These virtual machines share the performance of a single physical GPU. For some GPUs, only one mediated device can be assigned to a single guest. vGPU support is only available on selected NVIDIA GPUs.

  Example:

  A host has four GPUs. Each GPU can support up to 16 vGPUs, for a total of 64 vGPUs. Some possible vGPU assignments are:

  - one virtual machine with 64 vGPUs
  - 64 virtual machines, each with one vGPU
  - 32 virtual machines, each with one vGPU; eight virtual machines, each with two vGPUs; 4 virtual machines, each with four vGPUs

CHAPTER 1. GPU DEVICE PASSTHROUGH: ASSIGNING A HOST GPU TO A SINGLE VIRTUAL MACHINE

Red Hat Virtualization supports PCI VFIO, also called device passthrough, for some NVIDIA PCIe-based GPU devices as non-VGA graphics devices.

You can attach one or more host GPUs to a single virtual machine by passing through the host GPU to the virtual machine, in addition to one of the standard emulated graphics interfaces. The virtual machine uses the emulated graphics device for pre-boot and installation, and the GPU takes control when its graphics drivers are loaded.

For information on the exact number of host GPUs that you can pass through to a single virtual machine, see the NVIDIA website.

To assign a GPU to a virtual machine, follow the steps in these procedures:

1. Enable the I/O Memory Management Unit (IOMMU) on the host machine.
2. Detach the GPU from the host.
3. Attach the GPU to the guest.
4. Install GPU drivers on the guest.
5. Configure Xorg on the guest.

These steps are detailed below.

Prerequisites

- Your GPU device supports GPU passthrough mode.
- Your system is listed as a validated server hardware platform.
- Your host chipset supports Intel VT-d or AMD-Vi

For more information about supported hardware and software, see Validated Platforms in the *NVIDIA GPU Software Release Notes*.

1.1. ENABLING HOST IOMMU SUPPORT AND BLACKLISTING NOUVEAU

I/O Memory Management Unit (IOMMU) support on the host machine is necessary to use a GPU on a virtual machine.

Procedure

1. In the Administration Portal, click **Compute → Hosts**. Select a host and click **Edit**. The **Edit Hosts** pane appears.
2. Click the **Kernel** tab.
3. Check the **Hostdev Passthrough & SR-IOV** checkbox. This checkbox enables IOMMU support for a host with Intel VT-d or AMD Vi by adding **intel_iommu=on** or **amd_iommu=on** to the kernel command line.

4. Check the **Blacklist Nouveau** checkbox.
5. Click **OK**.
6. Select the host and click **Management → Maintenance**.
7. Click **Installation → Reinstall**.
8. After the reinstallation is finished, reboot the host machine.
9. When the host machine has rebooted, click **Management → Activate**.

1.2. DETACHING THE GPU FROM THE HOST

You cannot add the GPU to the virtual machine if the GPU is bound to the host kernel driver, so you must unbind the GPU device from the host before you can add it to the virtual machine. Host drivers often do not support dynamic unbinding of the GPU, so it is recommended to manually exclude the device from binding to the host drivers.

Procedure

1. On the host, identify the device slot name and IDs of the device by running the **lspci** command. In the following example, a graphics controller such as an NVIDIA Quadro or GRID card is used:

```
# lspci -Dnn | grep -i NVIDIA
0000:03:00.0 VGA compatible controller [0300]: NVIDIA Corporation GK104GL [Quadro
K4200] [10de:11b4] (rev a1)
0000:03:00.1 Audio device [0403]: NVIDIA Corporation GK104 HDMI Audio Controller
[10de:0e0a] (rev a1)
```

   The output shows that the NVIDIA GK104 device is installed. It has a graphics controller and an audio controller with the following properties:

   - The device slot name of the graphics controller is **0000:03:00.0**, and the vendor-id:device-id for the graphics controller are **10de:11b4**.
   - The device slot name of the audio controller is **0000:03:00.1**, and the vendor-id:device-id for the audio controller are **10de:0e0a**.

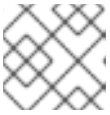
2. Prevent the host machine driver from using the GPU device. You can use a vendor-id:device-id with the pci-stub driver. To do this, append the **pci-stub.ids** option, with the vendor-id:device-id as its value, to the **GRUB_CMDLINX_LINUX** environment variable located in the **/etc/sysconfig/grub** configuration file, for example:

```
GRUB_CMDLINE_LINUX="crashkernel=auto resume=/dev/mapper/vg0-lv_swap
rd.lvm.lv=vg0/lv_root rd.lvm.lv=vg0/lv_swap rhgb quiet intel_iommu=on pci-
stub.ids=10de:11b4,10de:0e0a"
```

   When adding additional vendor IDs and device IDs for pci-stub, separate them with a comma.

3. Regenerate the boot loader configuration using grub2-mkconfig to include this option, as follows:

```
# grub2-mkconfig -o /etc/grub2.cfg
```



NOTE

When using a UEFI-based host, the target file should be **/etc/grub2-efi.cfg**.

4. Reboot the host machine.

5. Confirm that IOMMU is enabled, the host device is added to the list of pci-stub.ids, and Nouveau is blacklisted:

```
# cat /proc/cmdline
BOOT_IMAGE=(hd0,msdos1)/vmlinuz-4.18.0-147.el8.x86_64 root=/dev/mapper/vg0-lv_root
ro crashkernel=auto resume=/dev/mapper/vg0-lv_swap rd.lvm.lv=vg0/lv_root
rd.lvm.lv=vg0/lv_swap rhgb quiet intel_iommu=on (1)
pci-stub.ids=10de:11b4,10de:0e0a (2)
rdblacklist=nouveau (3)
```

(1) IOMMU is enabled

(2) the host device is added to the list of pci-stub.ids

(3) Nouveau is blacklisted

1.3. ATTACHING THE GPU TO A VIRTUAL MACHINE

After unbinding the GPU from host kernel driver, you can add it to the virtual machine and enable the correct driver.

Procedure

1. Follow the steps in Adding a Host Device to a Virtual Machine in the *Virtual Machine Management Guide*.

2. Run the virtual machine and log in to it.

3. Install the NVIDIA GPU driver on the virtual machine. For details, see Section 1.4, "Installing the GPU driver on the virtual machine".

4. Verify that the correct kernel driver is in use for the GPU with the **lspci -nnk** command. For example:

```
# lspci -nnk
00:07.0 VGA compatible controller [0300]: NVIDIA Corporation GK104GL [Quadro K4200]
[10de:11b4] (rev a1)
Subsystem: Hewlett-Packard Company Device [103c:1096]
Kernel driver in use: nvidia
Kernel modules: nouveau, nvidia_drm, nvidia
```

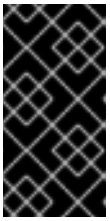
1.4. INSTALLING THE GPU DRIVER ON THE VIRTUAL MACHINE

Procedure

1. Run the virtual machine and connect to it using a serial console, such as SPICE or VNC.

2. Download the driver to the virtual machine. For information on getting the driver, see the Drivers page on the NVIDIA website.

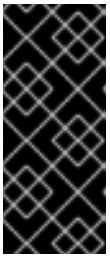
3. Install the GPU driver.



IMPORTANT

Linux only: When installing the driver on a Linux guest operating system, you are prompted to update xorg.conf. If you do not update xorg.conf during the installation, you need to update it manually. For more information, see Section 1.5, "Updating and Enabling xorg (Linux Virtual Machines)" .

4. After the driver finishes installing, reboot the machine. **For Windows virtual machines,** fully power off the guest from the Administration portal or the VM portal, not from within the guest operating system.



IMPORTANT

Windows only: Powering off the virtual machine from within the Windows guest operating system sometimes sends the virtual machine into hibernate mode, which does not completely clear the memory, possibly leading to subsequent problems. Using the Administration portal or the VM portal to power off the virtual machine forces it to fully clean the memory.

5. Connect a monitor to the host GPU output interface and run the virtual machine.

6. Set up NVIDIA vGPU guest software licensing for each vGPU and add the license credentials in the NVIDIA control panel. For more information, see How NVIDIA vGPU Software Licensing Is Enforced in the *NVIDIA Virtual GPU Software Documentation*.

1.5. UPDATING AND ENABLING XORG (LINUX VIRTUAL MACHINES)

Before you can use the GPU on the virtual machine, you need to update and enable xorg on the virtual machine. The NVIDIA driver installation should do this automatically. Check if xorg is updated and enabled by viewing **/etc/X11/xorg.conf**:

```
# cat /etc/X11/xorg.conf
```

The first two lines say if it was generated by NVIDIA. For example:

```
# cat /etc/X11/xorg.conf
# nvidia-xconfig: X configuration file generated by nvidia-xconfig
# nvidia-xconfig: version 390.87 (buildmeister@swio-display-x64-rhel04-14) Tue Aug 21 17:33:38
PDT 2018
```

Procedure

1. On the virtual machine, generate the **xorg.conf** file using following command:

```
# X -configure
```

2. Copy the **xorg.conf** file to **/etc/X11/xorg.conf** using the following command:

```
# cp /root/xorg.conf.new /etc/X11/xorg.conf
```

3. Reboot the virtual machine.

4. Verify that xorg is updated and enabled by viewing **/etc/X11/xorg.conf**:

```
# cat /etc/X11/xorg.conf
```

Search for the **Device** section. You should see an entry similar to the following section:

```
Section "Device"
    Identifier     "Device0"
    Driver         "nvidia"
    VendorName     "NVIDIA Corporation"
EndSection
```

The GPU is now assigned to the virtual machine.

1.6. REMOVING A HOST GPU FROM A VIRTUAL MACHINE

- For information on removing a host GPU from a virtual machine, see Removing Host Devices from a Virtual Machine in the *Virtual Machine Management Guide*.

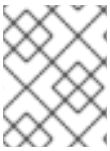
CHAPTER 2. ASSIGNING VIRTUAL GPUS

To set up NVIDIA vGPU devices, you need to:

1. Obtain and install the correct NVIDIA vGPU driver for your GPU device
2. Create mediated devices
3. Assign each mediated device to a virtual machine
4. Install guest drivers on each virtual machine.

The following procedures explain this process.

2.1. SETTING UP NVIDIA VGPU DEVICES ON THE HOST



NOTE

Before installing the NVIDIA vGPU driver on the guest operating system, you need to understand the licensing requirements and obtain the correct license credentials.

Prerequisites

- Your GPU device supports virtual GPU (vGPU) functionality.
- Your system is listed as a validated server hardware platform.

For more information about supported GPUs and validated platforms, see NVIDIA vGPU CERTIFIED SERVERS on www.nvidia.com.

Procedure

1. Download and install the NVIDIA driver for the host. For information on getting the driver, see the Drivers page on the NVIDIA website .
2. If the NVIDIA software installer did not create the **/etc/modprobe.d/nvidia-installer-disable-nouveau.conf** file, create it manually.
3. Open **/etc/modprobe.d/nvidia-installer-disable-nouveau.conf** file in a text editor and add the following lines to the end of the file:

```
blacklist nouveau
options nouveau modeset=0
```

4. Regenerate the initial ramdisk for the current kernel, then reboot:

```
# dracut --force
# reboot
```

Alternatively, if you need to use a prior supported kernel version with mediated devices, regenerate the initial ramdisk for all installed kernel versions:

```
# dracut --regenerate-all --force
# reboot
```

5. Check that the kernel loaded the **nvidia_vgpu_vfio** module:

```
# lsmod | grep nvidia_vgpu_vfio
```

6. Check that the **nvidia-vgpu-mgr.service** service is running:

```
# systemctl status nvidia-vgpu-mgr.service
```

For example:

```
# lsmod | grep nvidia_vgpu_vfio
nvidia_vgpu_vfio 45011 0
nvidia 14333621 10 nvidia_vgpu_vfio
mdev 20414 2 vfio_mdev,nvidia_vgpu_vfio
vfio 32695 3 vfio_mdev,nvidia_vgpu_vfio,vfio_iommu_type1
# systemctl status nvidia-vgpu-mgr.service
nvidia-vgpu-mgr.service - NVIDIA vGPU Manager Daemon
   Loaded: loaded (/usr/lib/systemd/system/nvidia-vgpu-mgr.service; enabled; vendor preset:
disabled)
   Active: active (running) since Fri 2018-03-16 10:17:36 CET; 5h 8min ago
 Main PID: 1553 (nvidia-vgpu-mgr)
 [...]
```

7. Get a list of available mdev types by entering the following lines in the terminal, or alternatively, in a script:

```
for device in /sys/class/mdev_bus/; do for mdev_type in
$device/mdev_supported_types/; do
    MDEV_TYPE=$(basename $mdev_type)
    DESCRIPTION=$(cat $mdev_type/description)
    NAME=$(cat $mdev_type/name)
    echo "mdev_type: $MDEV_TYPE --- description: $DESCRIPTION --- name: $NAME";
  done;
done | sort | uniq
```

NOTE

For **type-id** values for specific GPU devices, see Virtual GPU Types in the Virtual GPU software documentation. Notice that only Q-series NVIDIA vGPUs, such as GRID P4-2Q, are supported as mediated device GPU types on Linux virtual machines.

The output looks similar to the following:

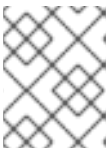
```
mdev_type: nvidia-11 --- description: num_heads=2, frl_config=45, framebuffer=512M,
max_resolution=2560x1600, max_instance=16 --- name: GRID M60-0B
mdev_type: nvidia-12 --- description: num_heads=2, frl_config=60, framebuffer=512M,
max_resolution=2560x1600, max_instance=16 --- name: GRID M60-0Q
...
mdev_type: nvidia-22 --- description: num_heads=4, frl_config=60, framebuffer=8192M,
max_resolution=4096x2160, max_instance=1 --- name: GRID M60-8Q
```

8. In the Administration Portal, click **Compute → Virtual Machines**. Select a virtual machine and click **Edit**. The **Edit Virtual Machine** dialog appears.

9. Click **Custom Properties**. If you do not see **Custom Properties**, click **Show Advanced Options** first.

10. In the **Custom Properties** dialog, click **Please select a key → mdev_type**. If you do not see **Please select a key**, click the **+** button.

11. In the text field that appears, enter the GPU type or types that you identified previously. For example: **nvidia-12**.You can add multiple vGPUs to a virtual machine using a comma-separated list. For example: **nvidia22,nvidia22**.



NOTE

Multiple vGPUs must be the same mdev type. You cannot, for example use **nvidia22,nvidia15**.

Now that you finished installing and configuring the GPU on the host, you can proceed to install and configure the vGPU on each virtual machine.

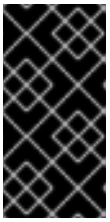
2.2. INSTALLING THE VGPU DRIVER ON THE VIRTUAL MACHINE

Procedure

1. Run the virtual machine and connect to it using a serial console, such as SPICE or VNC.

2. Download the driver to the virtual machine. For information on getting the driver, see the Drivers page on the NVIDIA website.

3. Install the vGPU driver, following the instructions in Installing the NVIDIA vGPU Software Graphics Driver in the *NVIDIA Virtual GPU software documentation*.



IMPORTANT

Linux only: When installing the driver on a Linux guest operating system, you are prompted to update xorg.conf. If you do not update xorg.conf during the installation, you need to update it manually. For more information, see Section 1.5, “Updating and Enabling xorg (Linux Virtual Machines)” .

4. After the driver finishes installing, reboot the machine. **For Windows virtual machines,** fully power off the guest from the Administration portal or the VM portal, not from within the guest operating system.

IMPORTANT

Windows only: Powering off the virtual machine from within the Windows guest operating system sometimes sends the virtual machine into hibernate mode, which does not completely clear the memory, possibly leading to subsequent problems. Using the Administration portal or the VM portal to power off the virtual machine forces it to fully clean the memory.

5. Run the virtual machine and connect to it using one of the supported remote desktop protocols,

such as Mechdyne TGX, and verify that the vGPU is recognized by opening the NVIDIA Control Panel. On Windows, you can alternatively open the Windows Device Manager. The vGPU should appear under **Display adapters**. For more information, see the NVIDIA vGPU Software Graphics Driver in the *NVIDIA Virtual GPU software documentation*.

6. Set up NVIDIA vGPU guest software licensing for each vGPU and add the license credentials in the NVIDIA control panel. For more information, see How NVIDIA vGPU Software Licensing Is Enforced in the *NVIDIA Virtual GPU Software Documentation*.

2.3. REMOVING NVIDIA VGPU DEVICES

To change the configuration of assigned vGPU mediated devices, the existing devices have to be removed from the assigned guests.

Procedure

1. From the Administration portal, click **Compute → Virtual Machines**.
2. Right-click the virtual machine and click **Power off**.
3. After the virtual machine is powered off, select the virtual machine and click **Edit**. The **Edit Virtual Machine** window opens.
4. On the **Custom Properties** tab, next to **mdev type**, click the minus **-** button and click **OK**.

2.4. MONITORING NVIDIA VGPUS

For NVIDIA vGPUS, to get info on the physical GPU and vGPU, you can use the NVIDIA System Management Interface by entering the **nvidia-smi** command on the host. For more information, see NVIDIA System Management Interface nvidia-smi in the *NVIDIA Virtual GPU Software Documentation*.

For example:

```
# nvidia-smi
Thu Nov  1 17:40:09 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.62                 Driver Version: 410.62                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|

|   0  Tesla M60           On   | 00000000:84:00.0 Off |                  Off |
| N/A   40C    P8    24W / 150W |   1034MiB /  8191MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla M60           On   | 00000000:85:00.0 Off |                  Off |
| N/A   33C    P8    23W / 150W |   8146MiB /  8191MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla M60           On   | 00000000:8B:00.0 Off |                  Off |
| N/A   34C    P8    24W / 150W |   8146MiB /  8191MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla M60           On   | 00000000:8C:00.0 Off |                  Off |
| N/A   45C    P8    24W / 150W |     18MiB /  8191MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
```

```
+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|

|    0     34432    C+G   vgpu                                    508MiB |
|    0     34718    C+G   vgpu                                    508MiB |
|    1     35032    C+G   vgpu                                   8128MiB |
|    2     35032    C+G   vgpu                                   8128MiB |
+-----------------------------------------------------------------------------+
```

2.5. REMOTE DESKTOP STREAMING SERVICES FOR NVIDIA VGPU

The following remote desktop streaming services have been successfully tested for use with the NVIDIA vGPU feature in RHEL 8:

- HP-RGS
- Mechdyne TGX - It is currently not possible to use Mechdyne TGX with Windows Server 2016 guests.
- NICE DCV - When using this streaming service, Red Hat recommends using fixed resolution settings, because using dynamic resolution in some cases results in a black screen.

NOTE

SPICE is not supported.

CHAPTER 3. RELATED INFORMATION

For further information on using NVIDA vGPU on RHEL with KVM, see:

- the NVIDIA GPU Software Release Notes .
- *NVIDIA Virtual GPU Software Documentation* at https://docs.nvidia.com.