proxmox gpu passthrough

Authors: fire1ce | Created: 2021-08-27 | Last update: 2023-07-13

Proxmox GPU Passthrough to VM

Introduction

GPU passthrough is a technology that allows the Linux kernel to present the internal PCI GPU directly to the virtual machine. The device behaves as if it were powered directly by the virtual machine, and the virtual machine detects the PCI device as if it were physically connected. We will cover how to enable GPU passthrough to a virtual machine in Proxmox VE.

Your mileage may vary depending on your hardware.

Proxmox Configuration for GPU Passthrough

The following examples uses `SSH` connection to the Proxmox server. The editor is `nano` but feel free to use any other editor. We will be editing the `grub` configuration file.

Find the PCI address of the GPU Device. The following command will show the PCI address of the GPU devices in Proxmox server:

```
lspci -nnv | grep VGA
```

Find the GPU you want to passthrough in result ts should be similar to this:

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce
RTX 2080 SUPER] [10de:1e81] (rev a1) (prog-if 00 [VGA controller])
```

What we are looking is the PCI address of the GPU device. In this case it's `01:00.0`. `01:00.0` is only a part of of a group of PCI devices on the GPU.
We can list all the devices in the group `01:00` by using the following command:

```
lspci -s 01:00
```

The usual output will include VGA Device and Audio Device. In my case, we have a USB

```
01:00.0 VGA compatible controller: NVIDIA Corporation TU104 [GeForce RTX 2080
SUPER] (rev a1)
01:00.1 Audio device: NVIDIA Corporation TU104 HD Audio Controller (rev a1)
01:00.2 USB controller: NVIDIA Corporation TU104 USB 3.1 Host Controller (rev
a1)
01:00.3 Serial bus controller [0c80]: NVIDIA Corporation TU104 USB Type-C UCSI
Controller (rev a1)
```

Now we need to get the id's of those devices. We can do this by using the following command:

```
lspci -s 01:00 -n
```

The output should look similar to this:

```
01:00.0 0300: 10de:1e81 (rev a1)
01:00.1 0403: 10de:10f8 (rev a1)
01:00.2 0c03: 10de:1ad8 (rev a1)
01:00.3 0c80: 10de:1ad9 (rev a1)
```

What we are looking are the pairs, we will use those id to split the PCI Group to separate devices.

```
10de:1e81,10de:10f8,10de:1ad8,10de:1ad9
```

Now it's time to edit the `grub` configuration file.

```
nano /etc/default/grub
```

Find the line that starts with `GRUB_CMDLINE_LINUX_DEFAULT` by default they should look like this:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
```

For Intel CPU

```
intel_iommu=on
```

For AMD CPU

```
amd_iommu=on
```

Then change it to look like this (Intel CPU example) and replace `vfio-pci.ids=` with the ids for the GPU you want to passthrough:

```
vfio-pci.ids=10de:1e81,10de:10f8,10de:1ad8,10de:1ad9
vfio_iommu_type1.allow_unsafe_interrupts=1 kvm.ignore_msrs=1
modprobe.blacklist=radeon,nouveau,nvidia,nvidiafb,nvidia-gpu"
```

Save the config changed and then update GRUB.

```
update-grub
```

Next we need to add `vfio` modules to allow PCI passthrough.

Edit the `/etc/modules` file.

```
nano /etc/modules
```

Add the following line to the end of the file:

```
# Modules required for PCI passthrough
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

Save and exit the editor.

Update configuration changes made in your /etc filesystem

```
update-initramfs -u -k all
```

Reboot Proxmox to apply the changes

Verify that IOMMU is enabled

```
dmesg | grep -e DMAR -e IOMMU
```

There should be a line that looks like `DMAR: IOMMU enabled`. If there is no output, something is wrong.

```
[0.000000] Warning: PCIe ACS overrides enabled; This may allow non-IOMMU
protected peer-to-peer DMA
[0.067203] DMAR: IOMMU enabled
[2.573920] pci 0000:00:00.2: AMD-Vi: IOMMU performance counters supported
[2.580393] pci 0000:00:00.2: AMD-Vi: Found IOMMU cap 0x40
[2.581776] perf/amd_iommu: Detected AMD IOMMU #0 (2 banks, 4 counters/bank).
```

Check that the GPU is in a separate IOMMU Group by using the following command:

```
#!/bin/bash
```

```
for g in $(find /sys/kernel/iommu_groups/* -maxdepth 0 -type d | sort -V); do
    echo "IOMMU Group ${g##*/}:"
    for d in $g/devices/*; do
        echo -e "\t$(lspci -nns ${d##*/})"
    done;
done;
```

Now your Proxmox host should be ready to GPU passthrough!

Windows Virtual Machine GPU Passthrough Configuration

For better results its recommend to use this Windwos 10/11 Virutal Machine configuration for proxmox.

Limitations & Workarounds

- In order for the GPU to to function properly in the VM, you must disable Proxmox's Virutal Display - Set it `none`.
- You will lose the ability to conect to the VM via Proxmox's Console.
- Display must be conected to the physical output of the GPU for the Windows Host to initialize the GPU properly.
- **You can use a HDMI Dummy Plug as a workaround - It will present itself as a HDMI Display to the Windows Host.**
- Make sure you have alternative way to connect to the VM for example via Remote Desktop (RDP).

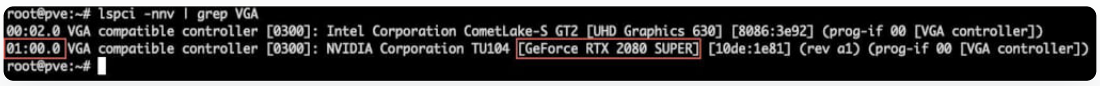
Find the PCI address of the GPU.

```
lspci -nnv | grep VGA
```

This should result in output similar to this:

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce
RTX 2080 SUPER] [10de:1e81] (rev a1) (prog-if 00 [VGA controller])
```

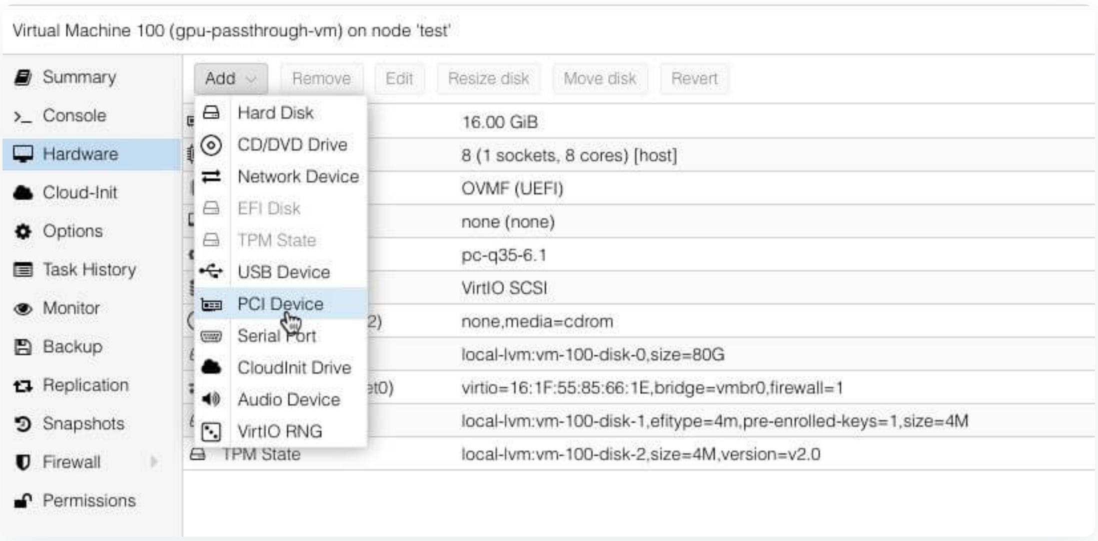
If you have multiple VGA, look for the one that has the `Intel` in the name.
Here, the PCI address of the GPU is `01:00.0`.



```
root@pve:~# lspci -nnv | grep VGA
00:02.0 VGA compatible controller [0300]: Intel Corporation CometLake-S GT2 [UHD Graphics 630] [8086:3e92] (prog-if 00 [VGA controller])
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce RTX 2080 SUPER] [10de:1e81] (rev a1) (prog-if 00 [VGA controller])
root@pve:~#
```

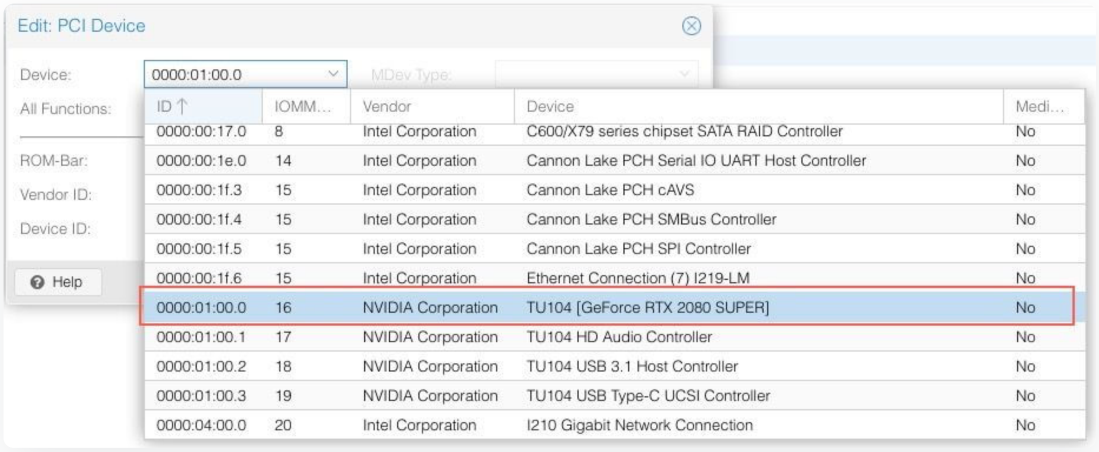
For best performance the VM should be configured the `Machine` type to q35.

Open the web gui and navigate to the `Hardware` tab of the VM you want to add a vGPU. Click `Add` above the device list and then choose `PCI Device`



	16.00 GiB
	8 (1 sockets, 8 cores) [host]
	OVMF (UEFI)
	none (none)
	pc-q35-6.1
	VirtIO SCSI
2)	none,media=cdrom
	local-lvm:vm-100-disk-0,size=80G
et0)	virtio=16:1F:55:85:66:1E,bridge=vmbr0,firewall=1
	local-lvm:vm-100-disk-1,efitype=4m,pre-enrolled-keys=1,size=4M
TPM State	local-lvm:vm-100-disk-2,size=4M,version=v2.0

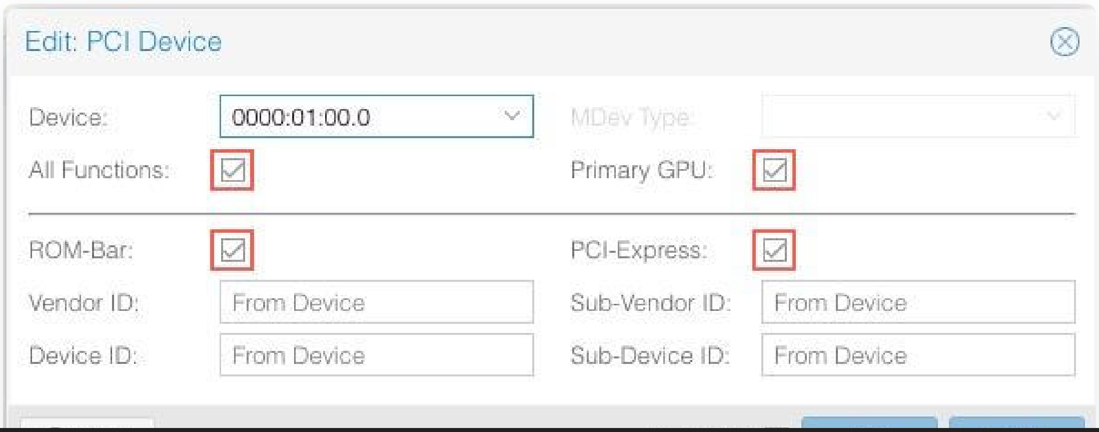
Open the `Device` dropdown and select the GPU, which you can find using it's PCI address. This list uses a different format for the PCI addresses id, `01:00.0` is listed as `0000:01:00.0`.



Device: 0000:01:00.0

ID ↑	IOMM...	Vendor	Device	Medi...
0000:00:17.0	8	Intel Corporation	C600/X79 series chipset SATA RAID Controller	No
0000:00:1e.0	14	Intel Corporation	Cannon Lake PCH Serial IO UART Host Controller	No
0000:00:1f.3	15	Intel Corporation	Cannon Lake PCH cAVS	No
0000:00:1f.4	15	Intel Corporation	Cannon Lake PCH SMBus Controller	No
0000:00:1f.5	15	Intel Corporation	Cannon Lake PCH SPI Controller	No
0000:00:1f.6	15	Intel Corporation	Ethernet Connection (7) I219-LM	No
0000:01:00.0	16	NVIDIA Corporation	TU104 [GeForce RTX 2080 SUPER]	No
0000:01:00.1	17	NVIDIA Corporation	TU104 HD Audio Controller	No
0000:01:00.2	18	NVIDIA Corporation	TU104 USB 3.1 Host Controller	No
0000:01:00.3	19	NVIDIA Corporation	TU104 USB Type-C UCSI Controller	No
0000:04:00.0	20	Intel Corporation	I210 Gigabit Network Connection	No

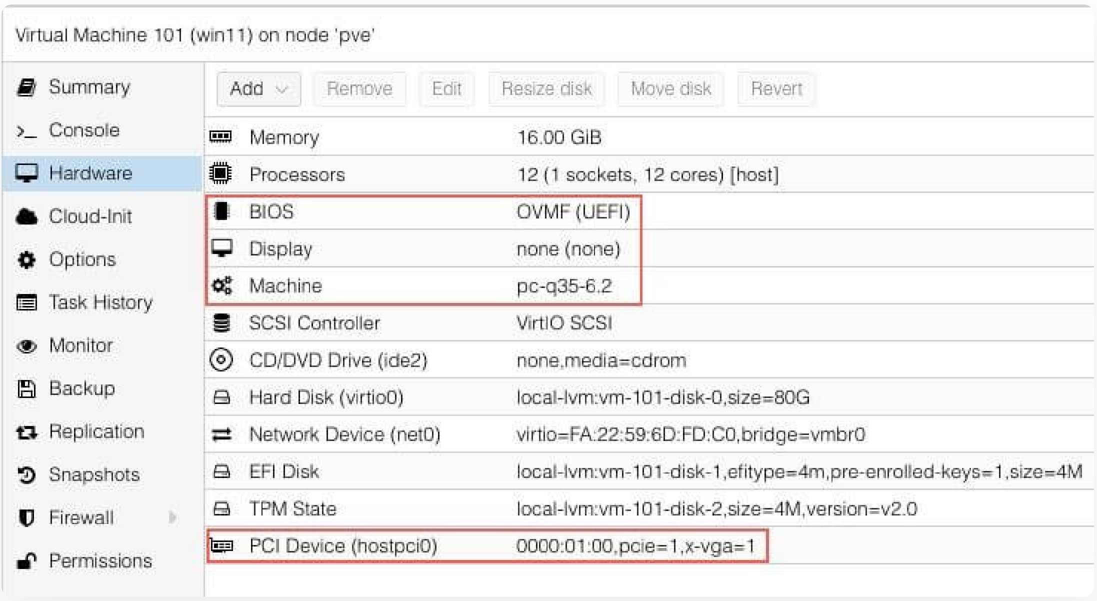
Select `All Functions`, `ROM-Bar`, `Primary GPU`, `PCI-Express` and then click `Add`.



Device:	0000:01:00.0	MDev Type:	
All Functions:	✓	Primary GPU:	✓
ROM-Bar:	✓	PCI-Express:	✓
Vendor ID:	From Device	Sub-Vendor ID:	From Device
Device ID:	From Device	Sub-Device ID:	From Device

Get 50% off Torguard VPN With Coupon: **all50torgourd**

The Windows Virtual Machine Proxmox Setting should look like this:

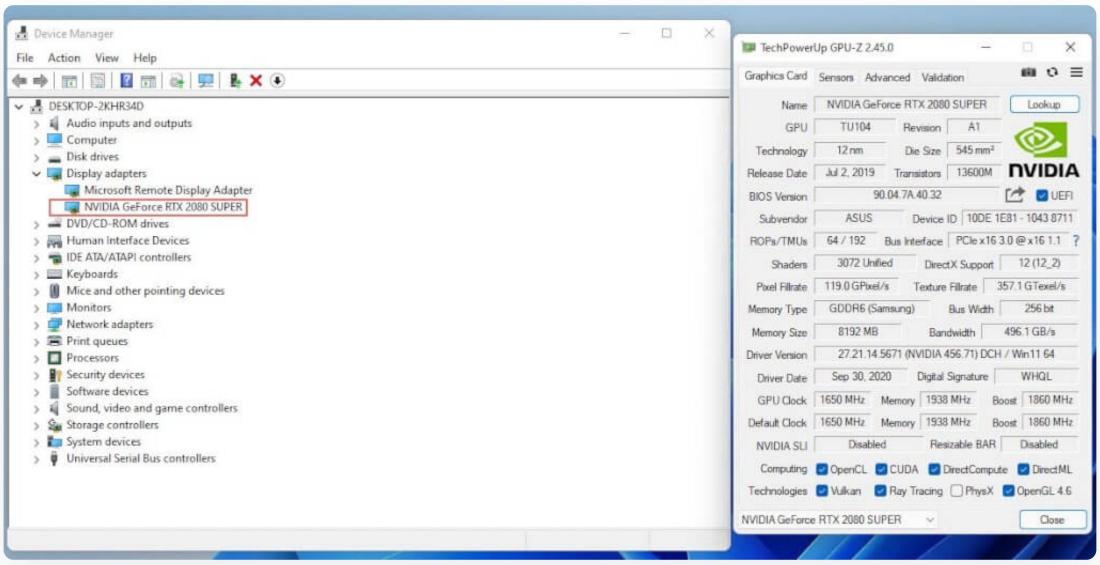


Memory	16.00 GiB
Processors	12 (1 sockets, 12 cores) [host]
BIOS	OVMF (UEFI)
Display	none (none)
Machine	pc-q35-6.2
SCSI Controller	VirtIO SCSI
CD/DVD Drive (ide2)	none,media=cdrom
Hard Disk (virtio0)	local-lvm:vm-101-disk-0,size=80G
Network Device (net0)	virtio=FA:22:59:6D:FD:C0,bridge=vmbr0
EFI Disk	local-lvm:vm-101-disk-1,efitype=4m,pre-enrolled-keys=1,size=4M
TPM State	local-lvm:vm-101-disk-2,size=4M,version=v2.0
PCI Device (hostpci0)	0000:01:00,pcie=1,x-vga=1

Power on the Windows Virtual Machine.

Connect to the VM via Remote Desktop (RDP) or any other remote access protocol you prefer. Install the latest version of GPU Driver for your GPU.

If all when well you should see the following output in `Device Manager` and GPU-Z:



```
DESKTOP-2KHR34D
  Audio inputs and outputs
  Computer
  Disk drives
  Display adapters
    Microsoft Remote Display Adapter
    NVIDIA GeForce RTX 2080 SUPER
  DVD/CD-ROM drives
  Human Interface Devices
  IDE ATA/ATAPI controllers
  Keyboards
  Mice and other pointing devices
  Monitors
  Network adapters
  Print queues
  Processors
  Security devices
  Software devices
  Sound, video and game controllers
  Storage controllers
  System devices
  Universal Serial Bus controllers
```

```
Name            NVIDIA GeForce RTX 2080 SUPER
GPU             TU104            Revision        A1
Technology      12 nm            Die Size        545 mm²
Release Date    Jul 2, 2019      Transistors     13600M
BIOS Version    90.04.7A.40.32                   UEFI
Subvendor       ASUS             Device ID       10DE 1E81 - 1043 8711
ROPs/TMUs       64 / 192         Bus Interface   PCIe x16 3.0 @ x16 1.1
Shaders         3072 Unified     DirectX Support 12 (12_2)
Pixel Fillrate  119.0 GPixel/s   Texture Fillrate 357.1 GTexel/s
Memory Type     GDDR6 (Samsung)  Bus Width       256 bit
Memory Size     8192 MB          Bandwidth       496.1 GB/s
Driver Version  27.21.14.5671 (NVIDIA 456.71) DCH / Win11 64
Driver Date     Sep 30, 2020     Digital Signature WHQL
GPU Clock       1650 MHz   Memory 1938 MHz   Boost 1860 MHz
Default Clock   1650 MHz   Memory 1938 MHz   Boost 1860 MHz
NVIDIA SLI      Disabled         Resizable BAR   Disabled
Computing       ✓ OpenCL  ✓ CUDA  ✓ DirectCompute  ✓ DirectML
Technologies    ✓ Vulkan  ✓ Ray Tracing  ☐ PhysX  ✓ OpenGL 4.6
NVIDIA GeForce RTX 2080 SUPER
```

That's it!

Linux Virtual Machine GPU Passthrough Configuration

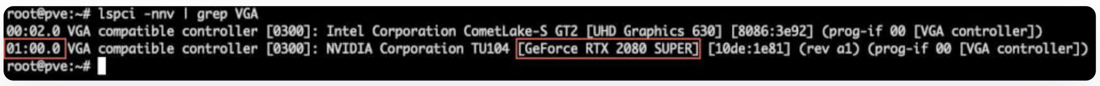
We will be using Ubuntu Server 20.04 LTS. for this guide.

```
lspci -nnv | grep VGA
```

This should result in output similar to this:

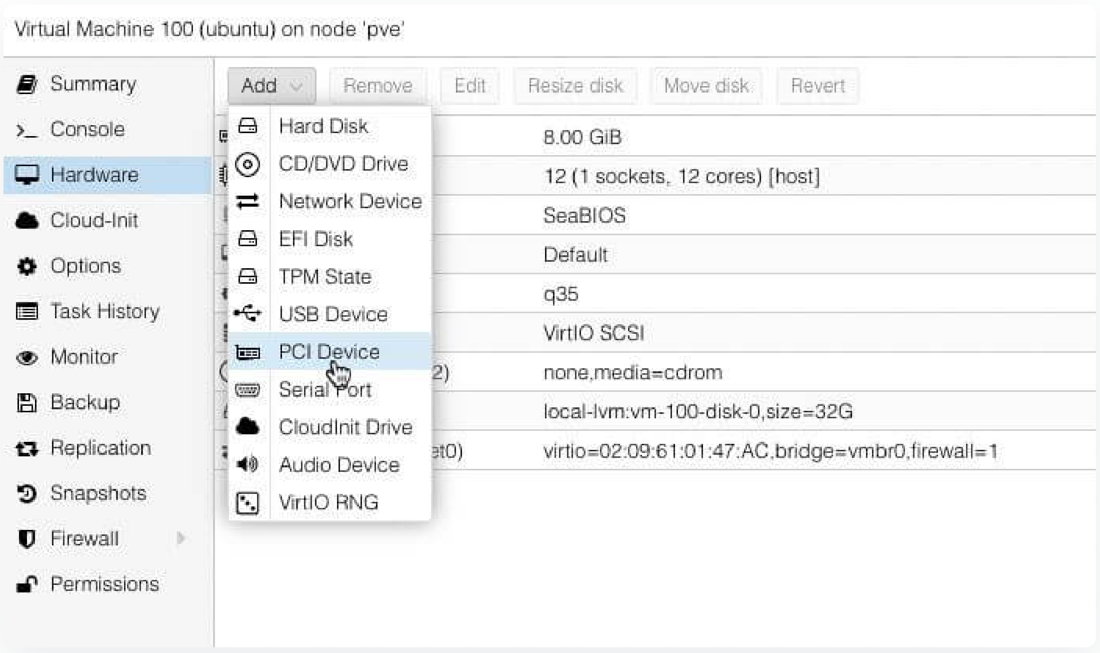
```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce
RTX 2080 SUPER] [10de:1e81] (rev a1) (prog-if 00 [VGA controller])
```

If you have multiple VGA, look for the one that has the `Intel` in the name. Here, the PCI address of the GPU is `01:00.0`.



```
root@pve:~# lspci -nnv | grep VGA
00:02.0 VGA compatible controller [0300]: Intel Corporation CometLake-S GT2 [UHD Graphics 630] [8086:3e92] (prog-if 00 [VGA controller])
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce RTX 2080 SUPER] [10de:1e81] (rev a1) (prog-if 00 [VGA controller])
root@pve:~# 
```

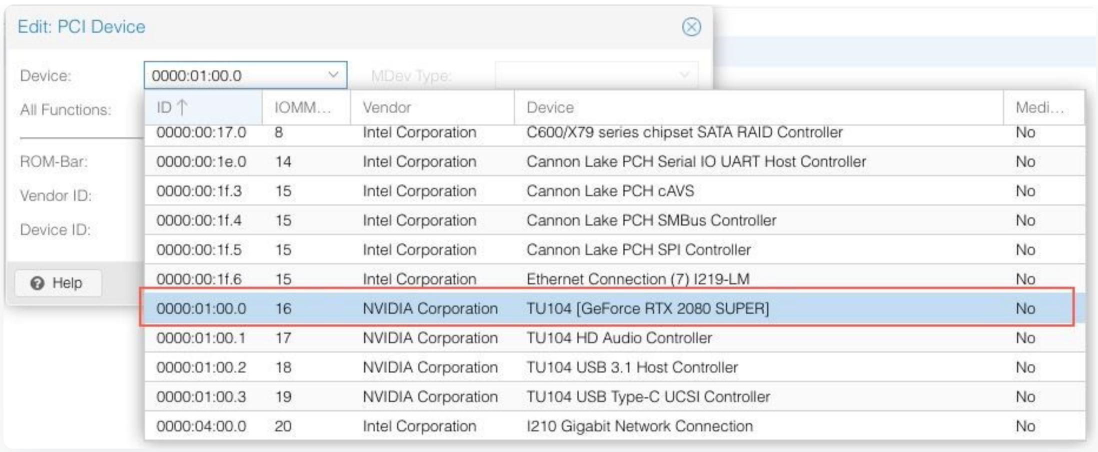
For best performance the VM should be configured the `Machine` type to q35.
This will allow the VM to utilize PCI-Express passthrough.



Virtual Machine 100 (ubuntu) on node 'pve'

8.00 GiB
12 (1 sockets, 12 cores) [host]
SeaBIOS
Default
q35
VirtIO SCSI
none,media=cdrom
local-lvm:vm-100-disk-0,size=32G
virtio=02:09:61:01:47:AC,bridge=vmbr0,firewall=1

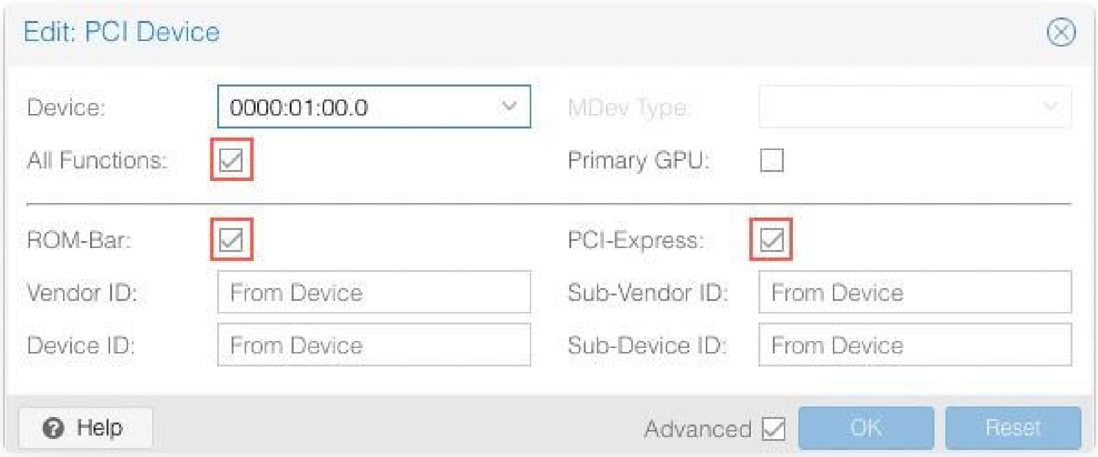
Open the `Device` dropdown and select the GPU, which you can find using it's PCI address. This list uses a different format for the PCI addresses id, `01:00.0` is listed as `0000:01:00.0`.



Device: 0000:01:00.0

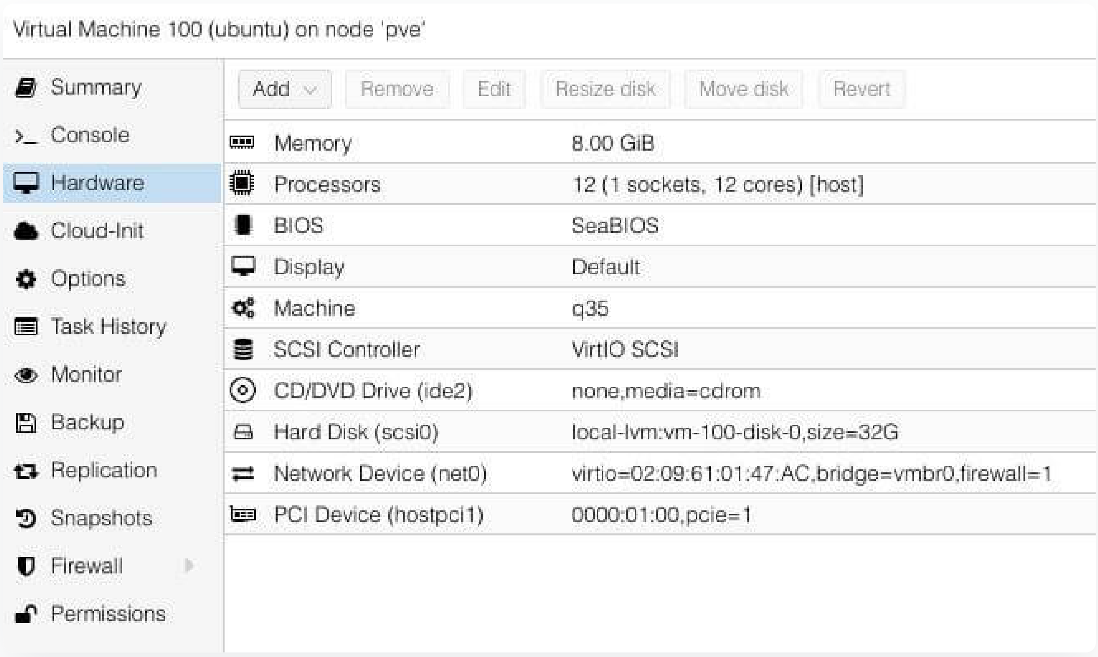
ID ↑	IOMM...	Vendor	Device	Medi...
0000:00:17.0	8	Intel Corporation	C600/X79 series chipset SATA RAID Controller	No
0000:00:1e.0	14	Intel Corporation	Cannon Lake PCH Serial IO UART Host Controller	No
0000:00:1f.3	15	Intel Corporation	Cannon Lake PCH cAVS	No
0000:00:1f.4	15	Intel Corporation	Cannon Lake PCH SMBus Controller	No
0000:00:1f.5	15	Intel Corporation	Cannon Lake PCH SPI Controller	No
0000:00:1f.6	15	Intel Corporation	Ethernet Connection (7) I219-LM	No
0000:01:00.0	16	NVIDIA Corporation	TU104 [GeForce RTX 2080 SUPER]	No
0000:01:00.1	17	NVIDIA Corporation	TU104 HD Audio Controller	No
0000:01:00.2	18	NVIDIA Corporation	TU104 USB 3.1 Host Controller	No
0000:01:00.3	19	NVIDIA Corporation	TU104 USB Type-C UCSI Controller	No
0000:04:00.0	20	Intel Corporation	I210 Gigabit Network Connection	No

Select `All Functions`, `ROM-Bar`, `PCI-Epress` and then click `Add`.



Device:	0000:01:00.0	MDev Type:	
All Functions:	✓	Primary GPU:	☐
ROM-Bar:	✓	PCI-Express:	✓
Vendor ID:	From Device	Sub-Vendor ID:	From Device
Device ID:	From Device	Sub-Device ID:	From Device

The Ubuntu Virtual Machine Proxmox Setting should look like this:



Memory	8.00 GiB
Processors	12 (1 sockets, 12 cores) [host]
BIOS	SeaBIOS
Display	Default
Machine	q35
SCSI Controller	VirtIO SCSI
CD/DVD Drive (ide2)	none,media=cdrom
Hard Disk (scsi0)	local-lvm:vm-100-disk-0,size=32G
Network Device (net0)	virtio=02:09:61:01:47:AC,bridge=vmbr0,firewall=1
PCI Device (hostpci1)	0000:01:00,pcie=1

Boot the VM. To test the GPU passthrough was successful, you can use the following

```
sudo lspci -nnv | grep VGA
```

The output should incliude the GPU:

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce
RTX 2080 SUPER] [10de:1e81] (rev a1) (prog-if 00 [VGA controller])
```

Now we need to install the GPU Driver. I'll be covering the installation of Nvidia Drivers in the next example.

Search for the latest Nvidia Driver for your GPU.

```
sudo apt search nvidia-driver
```

In the next step we will install the Nvidia Driver v535.

Note

--no-install-recommends is important for Headless Server. `nvidia-driver-535` will install xorg (GUI) `--no-install-recommends` flag will prevent the GUI from being installed.

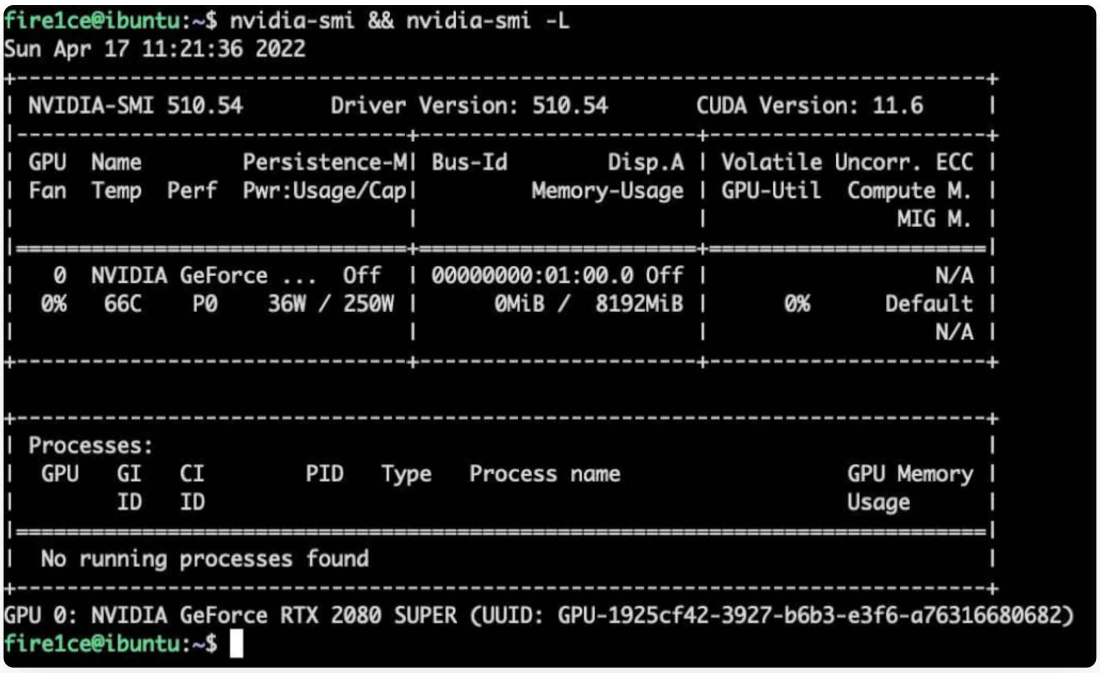
```
sudo apt install --no-install-recommends -y build-essential nvidia-driver-535
nvidia-headless-535 nvidia-utils-535 nvidia-cuda-toolkit
```

This will take a while to install. After the installation is complete, you should reboot the VM.

Now let's test the Driver initalization. Run the following command in the VM:

```
nvidia-smi && nvidia-smi -L
```

If all went well you should see the following output:



```
fire1ce@ibuntu:~$ nvidia-smi && nvidia-smi -L
Sun Apr 17 11:21:36 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.54       Driver Version: 510.54       CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:01:00.0 Off |                  N/A |
|  0%   66C    P0    36W / 250W |      0MiB /  8192MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
GPU 0: NVIDIA GeForce RTX 2080 SUPER (UUID: GPU-1925cf42-3927-b6b3-e3f6-a76316680682)
fire1ce@ibuntu:~$ 
```

That's it! You should now be able to use the GPU for hardware acceleration inside the VM.

Debug

Dbug Messages - Shows Hardware initialization and errors

```
dmesg -w
```

Display PCI devices information

```
lspci
```

Display Driver in use for PCI devices

```
lspci -k
```

Display IOMMU Groups the PCI devices are assigned to

```
#!/bin/bash
shopt -s nullglob
for g in $(find /sys/kernel/iommu_groups/* -maxdepth 0 -type d | sort -V); do
    echo "IOMMU Group ${g##*/}:"
    for d in $g/devices/*; do
        echo -e "\t$(lspci -nns ${d##*/})"
    done;
done;
```

Reboot Proxmox to apply the changes

Get 50% off Torguard VPN With Coupon: **all50torgourd**

Comments

0 reactions

1 comment *– powered by giscus*

Oldest Newest

Get 50% off Torguard VPN With Coupon: **all50torgourd**