proxmox igpu passthrough

Authors: fire1ce, Barry Staes, jmole | Created: 2022-02-04 | Last update: 2022-12-22

iGPU Passthrough to VM (Intel Integrated Graphics)

Introduction

Intel Integrated Graphics (iGPU) is a GPU that is integrated into the CPU. The GPU is a part of the CPU and is used to render graphics. Proxmox may be configured to use iGPU passthrough to VM to allow the VM to use the iGPU for hardware acceleration for example using video encoding/decoding and Transcoding for series like Plex and Emby. This guide will show you how to configure Proxmox to use iGPU passthrough to VM.

Your mileage may vary depending on your hardware. The following guide was tested with Intel Gen8 CPU.

There are two ways to use iGPU passthrough to VM. The first way is to use the `Full iGPU Passthrough` to VM. The second way is to use the `iGPU GVT-g` technology which allows as to split the iGPU into two parts. We will be covering the `Full iGPU Passthrough`. If you want to use the split `iGPU GVT-g Passthrough` you can find the guide here.

Proxmox Configuration for iGPU Full 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.

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"
```

We want to allow `passthrough` and `Blacklists` known graphics drivers to prevent

Warning

You will loose the ability to use the onboard graphics card to access the Proxmox's console since Proxmox won't be able to use the Intel's gpu

Your `GRUB_CMDLINE_LINUX_DEFAULT` should look like this:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt
pcie_acs_override=downstream,multifunction initcall_blacklist=sysfb_init
video=simplefb:off video=vesafb:off video=efifb:off video=vesa:off
disable_vga=1 vfio_iommu_type1.allow_unsafe_interrupts=1 kvm.ignore_msrs=1
modprobe.blacklist=radeon,nouveau,nvidia,nvidiafb,nvidia-
gpu,snd_hda_intel,snd_hda_codec_hdmi,i915"
```

Note

This will blacklist most of the graphics drivers from proxmox. If you have a specific driver you need to use for Proxmox Host you need to remove it from `modprobe.blacklist`

Save and exit the editor.

Update the grub configuration to apply the changes the next time the system boots.

```
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
```

Update configuration changes made in your /etc filesystem

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

Save and exit the editor.

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

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).
```

Windows Virtual Machine iGPU Passthrough Configuration

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

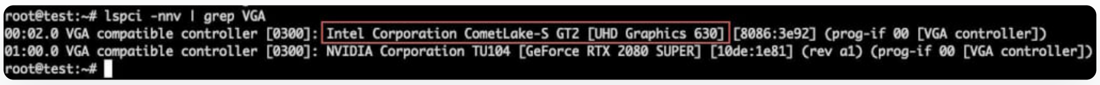
Find the PCI address of the iGPU.

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

This should result in output similar to this:

```
00:02.0 VGA compatible controller [0300]: Intel Corporation CometLake-S GT2
[UHD Graphics 630] [8086:3e92] (prog-if 00 [VGA controller])
```

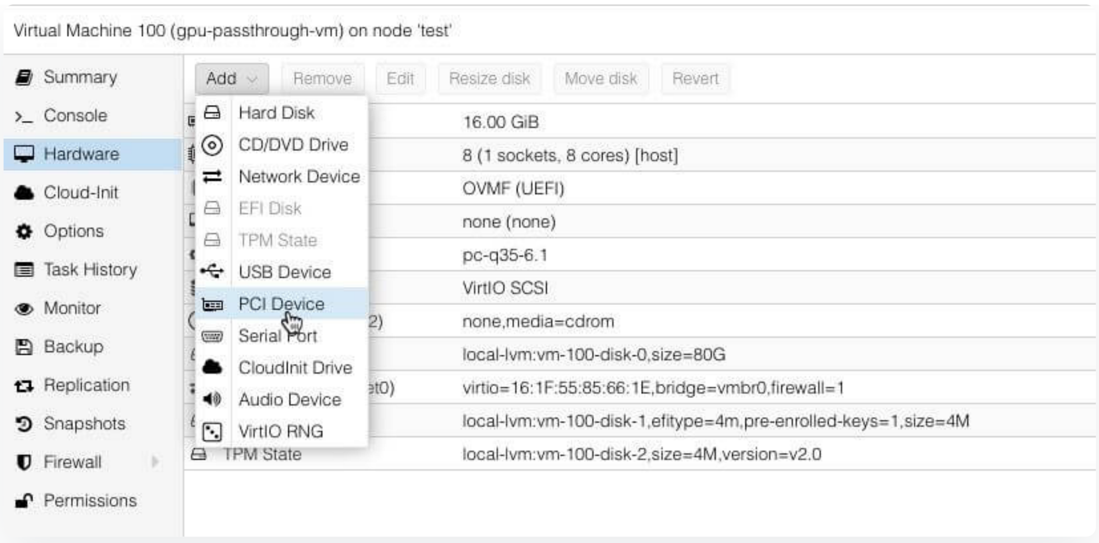
If you have multiple VGA, look for the one that has the `Intel` in the name.
Here, the PCI address of the iGPU is `00:02.0`.



```
root@test:~# 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@test:~#
```

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

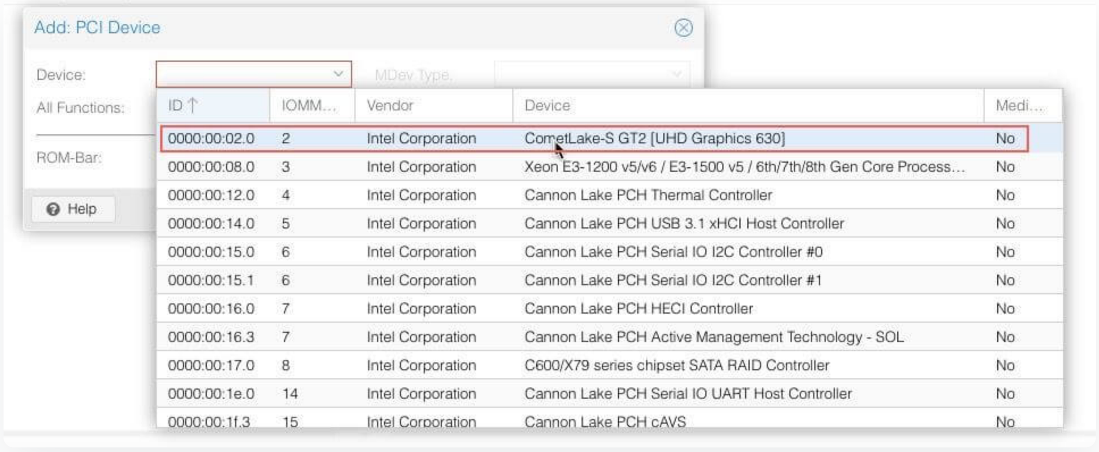
Open the web gui and navigate to the `Hardware` tab of the VM you want to add a vGPU.
Click `Add` above the device list and then choose `PCI Device`



Virtual Machine 100 (gpu-passthrough-vm) on node 'test'

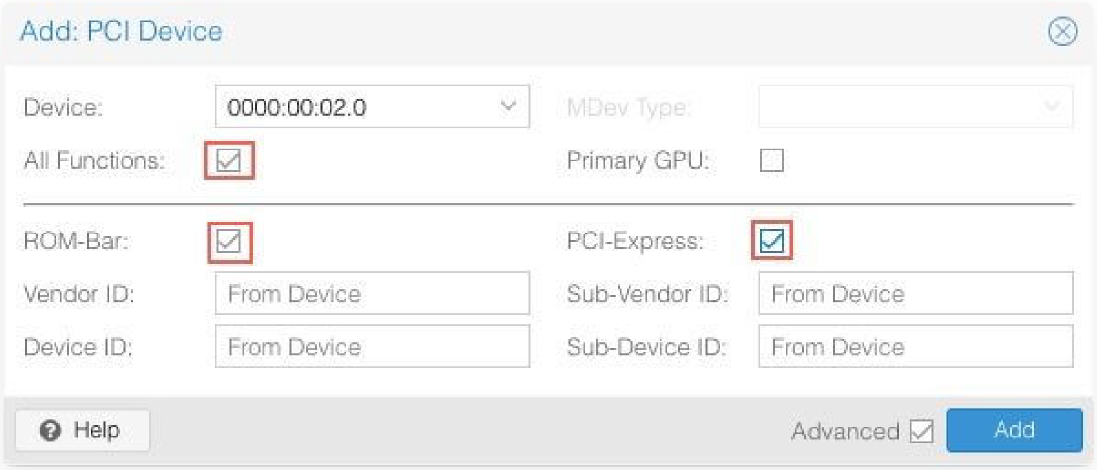
	16.00 GiB
	8 (1 sockets, 8 cores) [host]
	OVMF (UEFI)
	none (none)
	pc-q35-6.1
	VirtIO SCSI
2)	none,media=cdrom
	local-lvm:vm-100-disk-0,size=80G
et0)	virtio=16:1F:55:85:66:1E,bridge=vmbr0,firewall=1
	local-lvm:vm-100-disk-1,efitype=4m,pre-enrolled-keys=1,size=4M
TPM State	local-lvm:vm-100-disk-2,size=4M,version=v2.0

Open the `Device` dropdown and select the iGPU, which you can find using it's PCI address. This list uses a different format for the PCI addresses id, `00:02.0` is listed as `0000:00:02.0`.



ID ↑	IOMM...	Vendor	Device	Medi...
0000:00:02.0	2	Intel Corporation	CometLake-S GT2 [UHD Graphics 630]	No
0000:00:08.0	3	Intel Corporation	Xeon E3-1200 v5/v6 / E3-1500 v5 / 6th/7th/8th Gen Core Process...	No
0000:00:12.0	4	Intel Corporation	Cannon Lake PCH Thermal Controller	No
0000:00:14.0	5	Intel Corporation	Cannon Lake PCH USB 3.1 xHCI Host Controller	No
0000:00:15.0	6	Intel Corporation	Cannon Lake PCH Serial IO I2C Controller #0	No
0000:00:15.1	6	Intel Corporation	Cannon Lake PCH Serial IO I2C Controller #1	No
0000:00:16.0	7	Intel Corporation	Cannon Lake PCH HECI Controller	No
0000:00:16.3	7	Intel Corporation	Cannon Lake PCH Active Management Technology - SOL	No
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

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



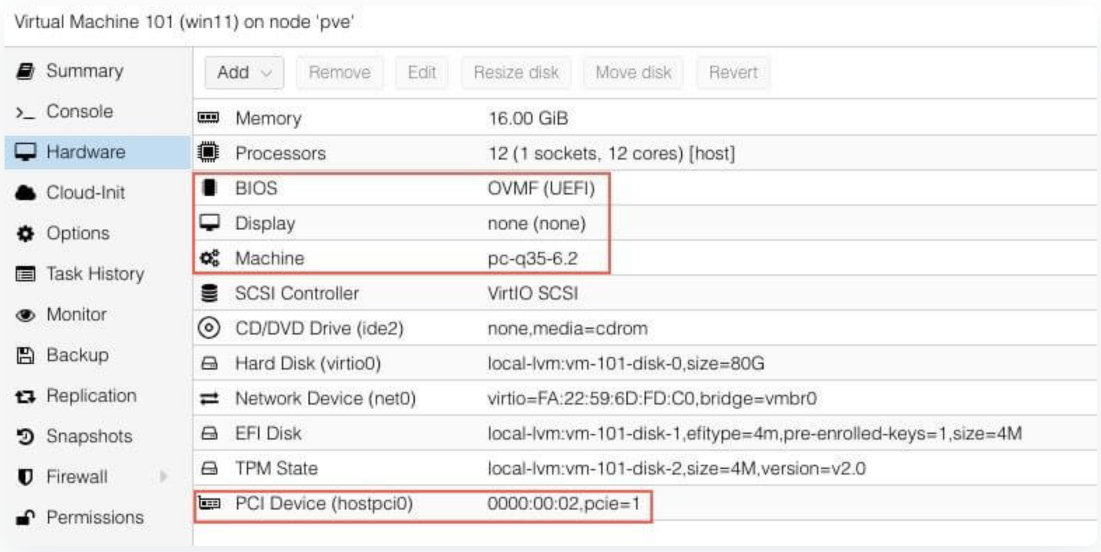
Device:	0000:00:02.0	MDev Type:	
All Functions:	✓	Primary GPU:	☐
ROM-Bar:	✓	PCI-Express:	✓
Vendor ID:	From Device	Sub-Vendor ID:	From Device
Device ID:	From Device	Sub-Device ID:	From Device
		Advanced	✓

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

Tip

I've found that the most consistent way to utilize the GPU acceleration is to disable Proxmox's Virtual Graphics card of the vm. The drawback of disabling the Virtual Graphics card is that it will not be able to access the vm via proxmox's vnc console. The workaround is to enable Remote Desktop (RDP) on the VM before disabling the Virtual Graphics card and accessing the VM via RDP or use any other remove desktop client. If you loose the ability to access the VM via RDP you can temporarily remove the GPU PCI Device and re-enable the virtual graphics card

The Windows Virtual Machine Proxmox Setting should look like this:

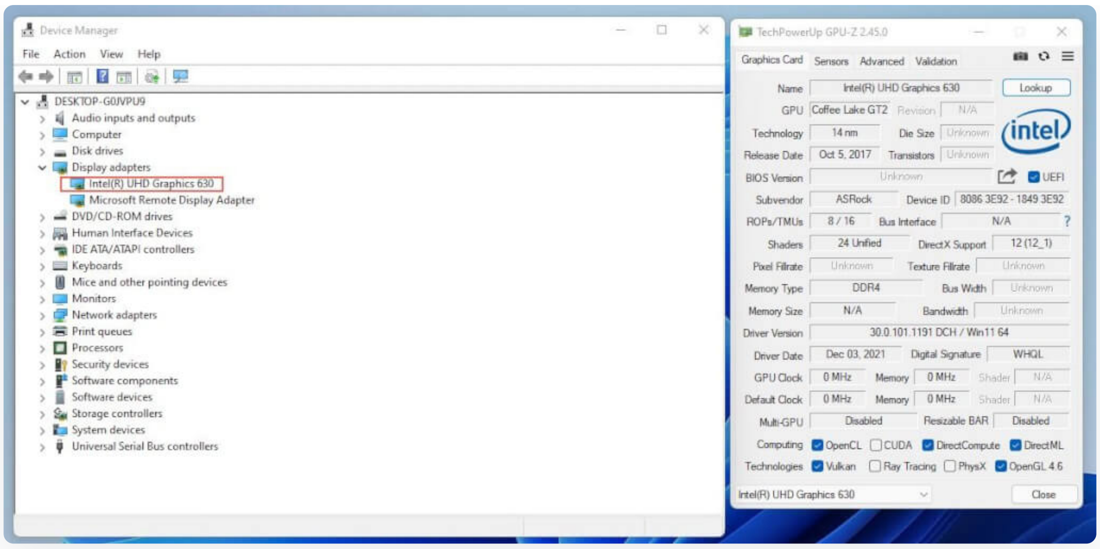


Memory	16.00 GiB
Processors	12 (1 sockets, 12 cores) [host]
BIOS	OVMF (UEFI)
Display	none (none)
Machine	pc-q35-6.2
SCSI Controller	VirtIO SCSI
CD/DVD Drive (ide2)	none,media=cdrom
Hard Disk (virtio0)	local-lvm:vm-101-disk-0,size=80G
Network Device (net0)	virtio=FA:22:59:6D:FD:C0,bridge=vmbr0
EFI Disk	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)	0000:00:02,pcie=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 Intel's Graphics Driver or use the Intel Driver & Support Assistant installer.

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



DESKTOP-G0JVPU9
- Audio inputs and outputs
- Computer
- Disk drives
- Display adapters
  - Intel(R) UHD Graphics 630
  - Microsoft Remote Display Adapter
- 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 components
- Software devices
- Storage controllers
- System devices
- Universal Serial Bus controllers

Name	Intel(R) UHD Graphics 630				
GPU	Coffee Lake GT2	Revision	N/A		
Technology	14 nm	Die Size	Unknown		
Release Date	Oct 5, 2017	Transistors	Unknown		
BIOS Version	Unknown	✓ UEFI			
Subvendor	ASRock	Device ID	8086 3E92 - 1849 3E92		
ROPs/TMUs	8 / 16	Bus Interface	N/A		
Shaders	24 Unified	DirectX Support	12 (12_1)		
Pixel Fillrate	Unknown	Texture Fillrate	Unknown		
Memory Type	DDR4	Bus Width	Unknown		
Memory Size	N/A	Bandwidth	Unknown		
Driver Version	30.0.101.1191 DCH / Win11 64				
Driver Date	Dec 03, 2021	Digital Signature	WHQL		
GPU Clock	0 MHz	Memory	0 MHz	Shader	N/A
Default Clock	0 MHz	Memory	0 MHz	Shader	N/A
Multi-GPU	Disabled	Resizable BAR	Disabled		
Computing	✓ OpenCL	☐ CUDA	✓ DirectCompute	✓ DirectML	
Technologies	✓ Vulkan	☐ Ray Tracing	☐ PhysX	✓ OpenGL 4.6	

Intel(R) UHD Graphics 630

That's it!

Linux Virtual Machine iGPU Passthrough Configuration

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

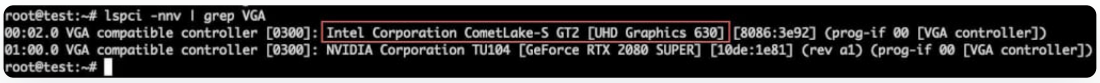
From Proxmox Terminal find the PCI address of the iGPU.

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

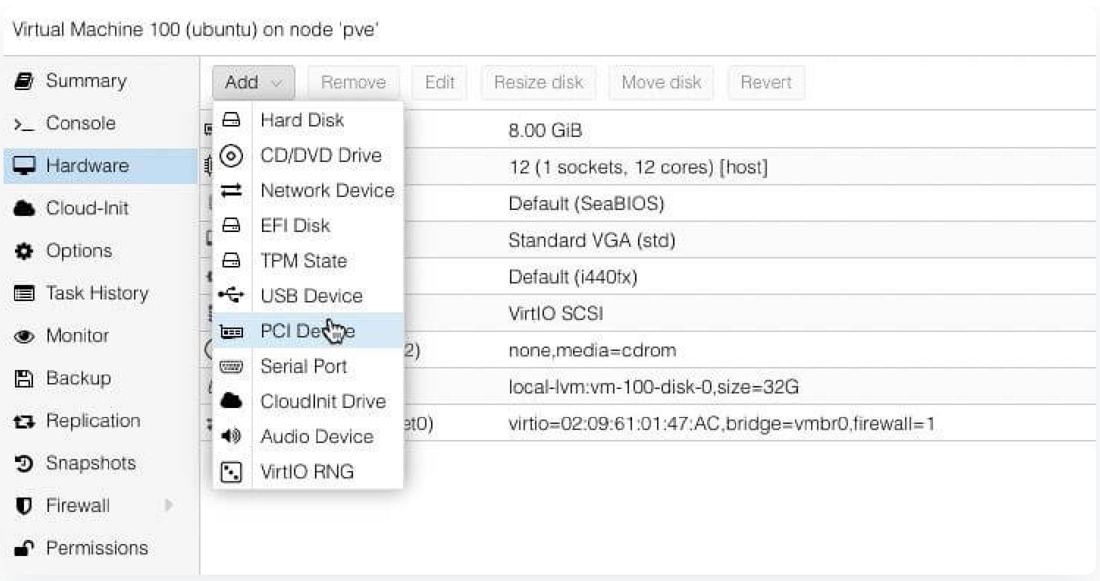
This should result in output similar to this:

```
00:02.0 VGA compatible controller [0300]: Intel Corporation CometLake-S GT2
[UHD Graphics 630] [8086:3e92] (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 iGPU is `00:02.0`.

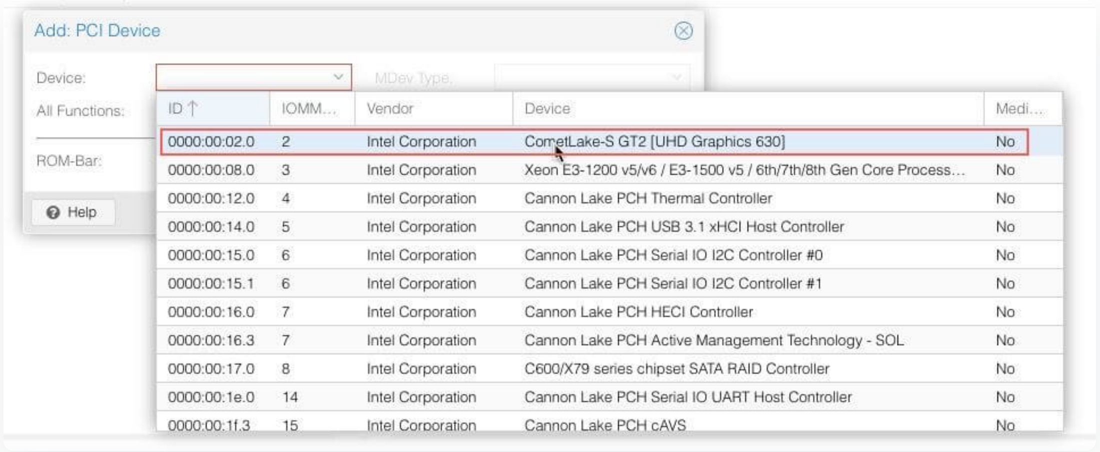


```
root@test:~# lspci -nnv | grep VGA
00:02.0 VGA compatible controller [0300]: Intel Corporation CometLake-S GT2 [UHD Graphics 630] [8086:3e92] (prog-if 00 [VGA controller])
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation TU104 [GeForce RTX 2080 SUPER] [10de:1e81] (rev a1) (prog-if 00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



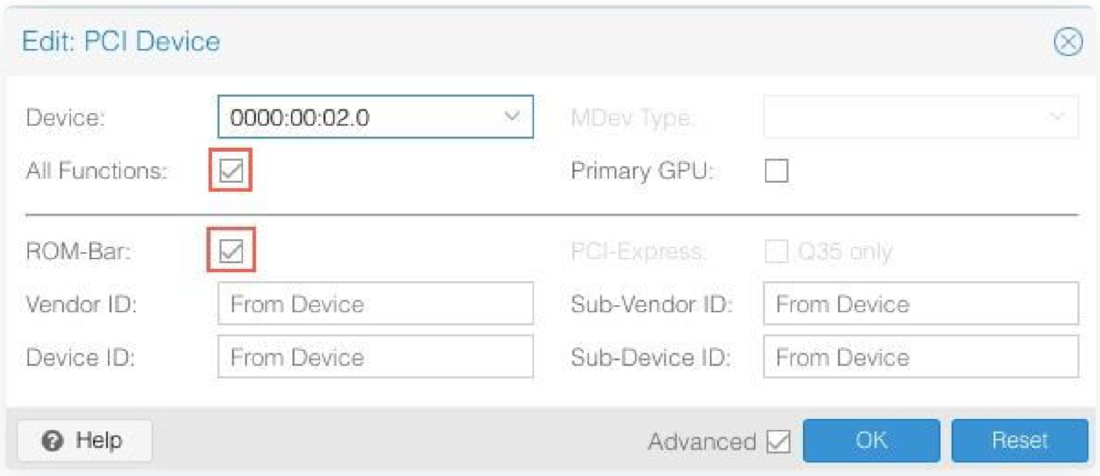
8.00 GiB
12 (1 sockets, 12 cores) [host]
Default (SeaBIOS)
Standard VGA (std)
Default (i440fx)
VirtIO SCSI
2) none,media=cdrom
local-lvm:vm-100-disk-0,size=32G
et0) virtio=02:09:61:01:47:AC,bridge=vmbr0,firewall=1

Open the `Device` dropdown and select the iGPU, which you can find using it's PCI address. This list uses a different format for the PCI addresses id, `00:02.0` is listed as `0000:00:02.0`.

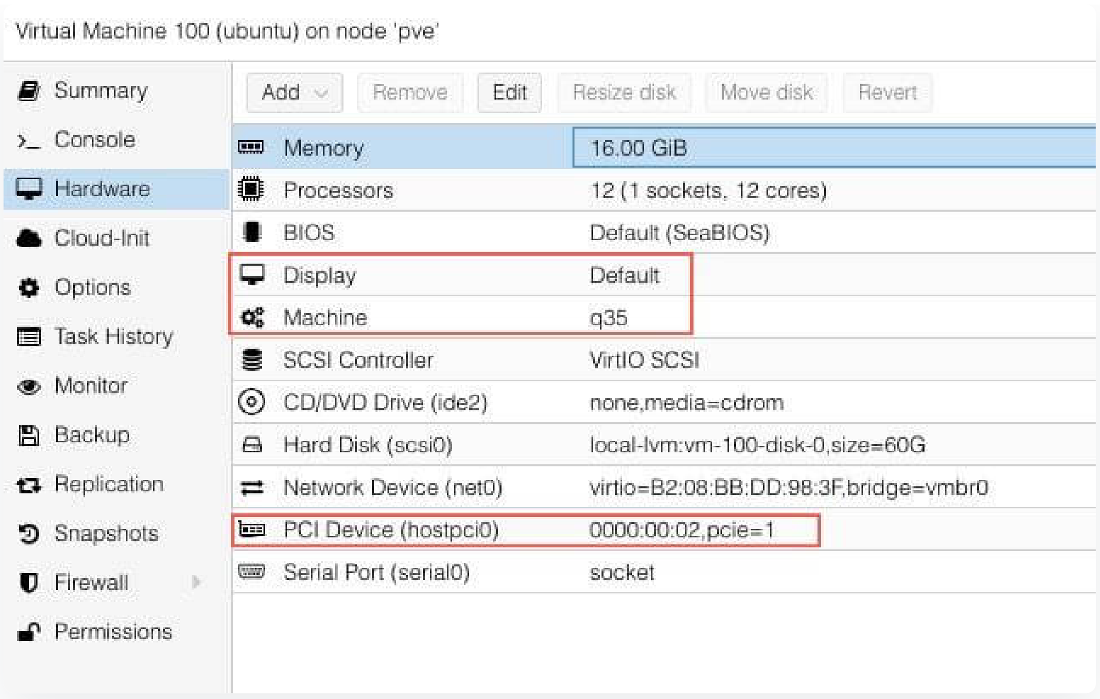


ID ↑	IOMM...	Vendor	Device	Medi...
0000:00:02.0	2	Intel Corporation	CometLake-S GT2 [UHD Graphics 630]	No
0000:00:08.0	3	Intel Corporation	Xeon E3-1200 v5/v6 / E3-1500 v5 / 6th/7th/8th Gen Core Process...	No
0000:00:12.0	4	Intel Corporation	Cannon Lake PCH Thermal Controller	No
0000:00:14.0	5	Intel Corporation	Cannon Lake PCH USB 3.1 xHCI Host Controller	No
0000:00:15.0	6	Intel Corporation	Cannon Lake PCH Serial IO I2C Controller #0	No
0000:00:15.1	6	Intel Corporation	Cannon Lake PCH Serial IO I2C Controller #1	No
0000:00:16.0	7	Intel Corporation	Cannon Lake PCH HECI Controller	No
0000:00:16.3	7	Intel Corporation	Cannon Lake PCH Active Management Technology - SOL	No
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

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



Device:	0000:00:02.0	MDev Type:	
All Functions:	☑	Primary GPU:	☐
ROM-Bar:	☑	PCI-Express:	☐ Q35 only
Vendor ID:	From Device	Sub-Vendor ID:	From Device
Device ID:	From Device	Sub-Device ID:	From Device



Memory	16.00 GiB
Processors	12 (1 sockets, 12 cores)
BIOS	Default (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=60G
Network Device (net0)	virtio=B2:08:BB:DD:98:3F,bridge=vmbr0
PCI Device (hostpci0)	0000:00:02,pcie=1
Serial Port (serial0)	socket

Boot the VM. To test the iGPU passthrough was successful, you can use the following command:

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

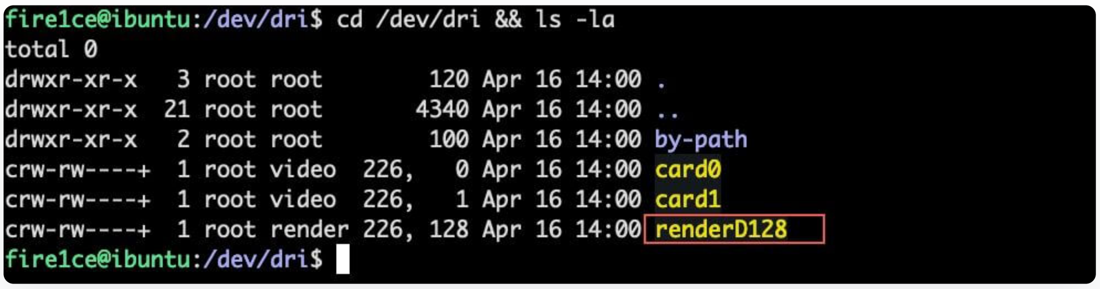
The output should include the Intel iGPU:

```
00:10.0 VGA compatible controller [0300]: Intel Corporation UHD Graphics 630
(Desktop) [8086:3e92] (prog-if 00 [VGA controller])
```

Now we need to check if the GPU's Driver initalization is working.

```
cd /dev/dri && ls -la
```

The output should include the `renderD128`



```
fire1ce@ibuntu:/dev/dri$ cd /dev/dri && ls -la
total 0
drwxr-xr-x   3 root root        120 Apr 16 14:00 .
drwxr-xr-x  21 root root       4340 Apr 16 14:00 ..
drwxr-xr-x   2 root root        100 Apr 16 14:00 by-path
crw-rw----+  1 root video  226,   0 Apr 16 14:00 card0
crw-rw----+  1 root video  226,   1 Apr 16 14:00 card1
crw-rw----+  1 root render 226, 128 Apr 16 14:00 renderD128
fire1ce@ibuntu:/dev/dri$
```

That's it! You should now be able to use the iGPU for hardware acceleration inside the VM and still have proxmox's output on the screen.

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;
```

Comments

0 reactions

14 comments · 16 replies *– powered by giscus*

Oldest Newest