Proxmox 6.2 GPU Passthrough Tutorial

Hardware

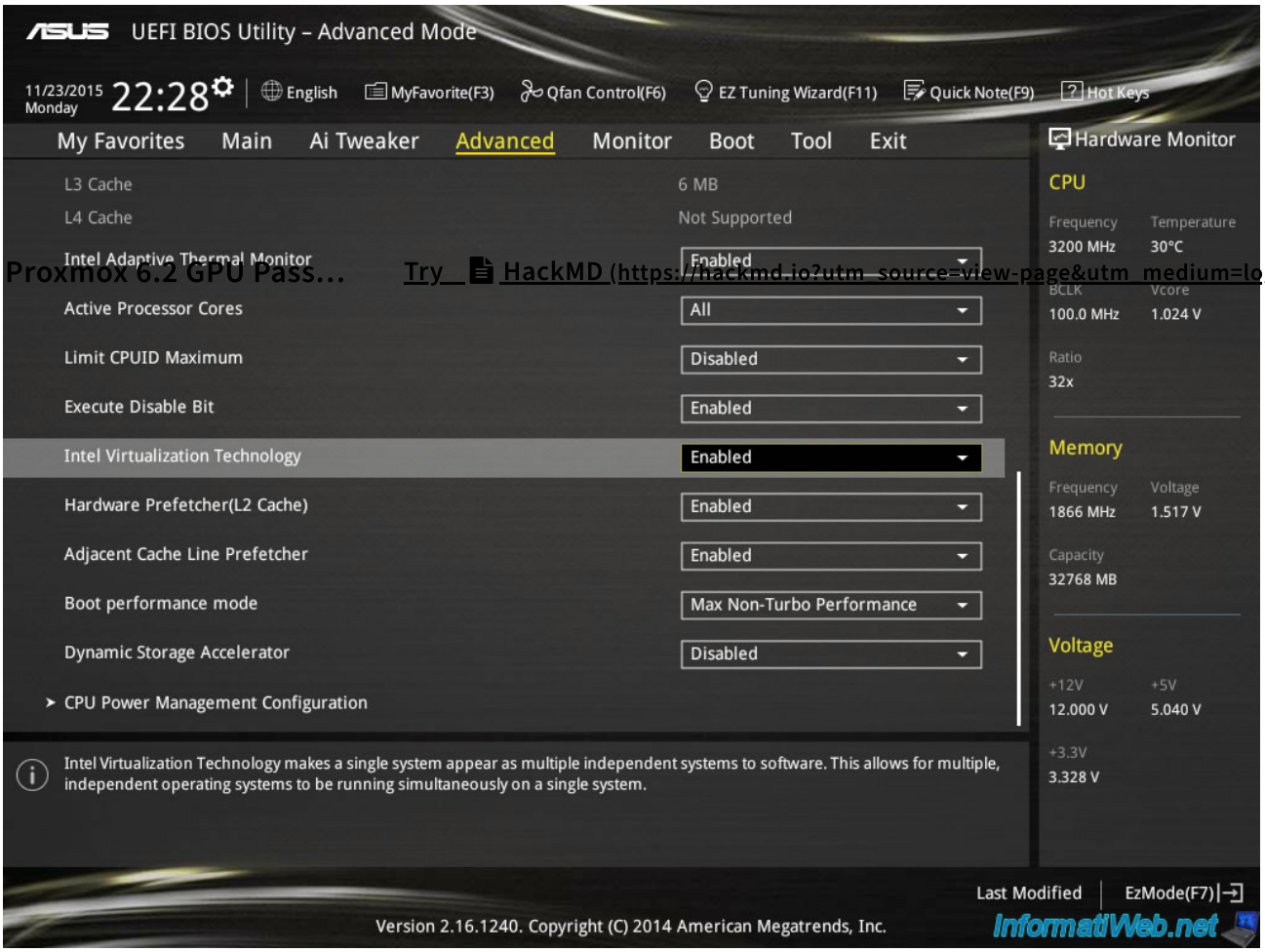
MB: ASUS TUF GAMING B460-PLUS
CPU: Intel i7-10700
GPU: NVIDIA GTX-1060

BIOS Config

Your hardware needs to support IOMMU (I/O Memory Management Unit) interrupt remapping, this includes the CPU and the mainboard.

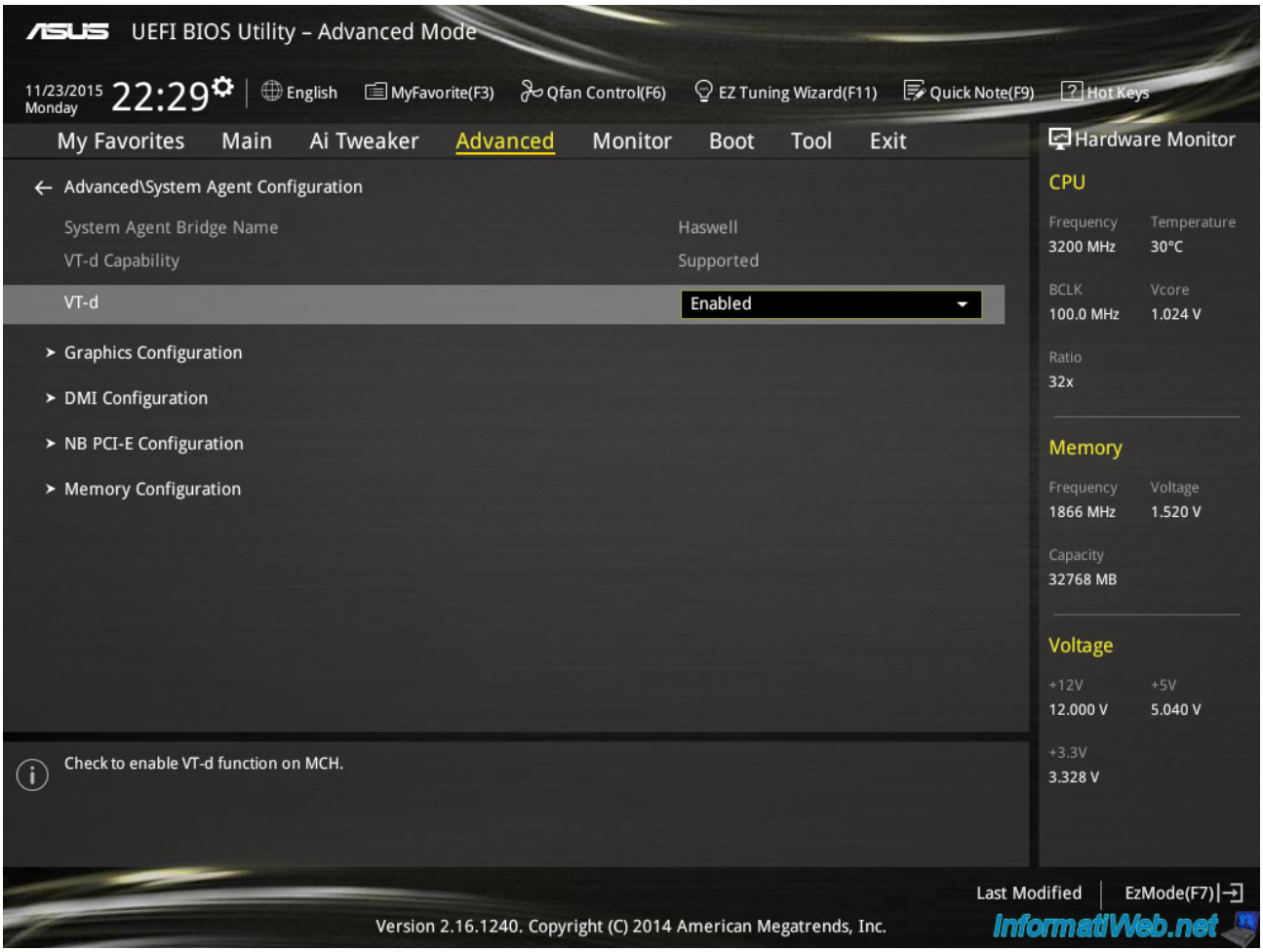
A. Enable VT-x

In the Asus UEFI BIOS, this feature is in “Advanced -> CPU configuration” and is named “Intel Virtualization Technology”.



B. Enable VT-d

Then, if your motherboard supports it, you will find the “VT-d” option that matches IOMMU in “Advanced -> System Agent Configuration” or “Advanced -> North Bridge”.



PS. SR-IOV option can also be found on this MB/BIOS, but will not be used in this tutorial.

Host Config

A. Load Required Modules and Block GPU Drivers

Load VFIO Modules

Add to `/etc/modules`

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

Block GPU Drivers

Block the original GPU drivers for attaching `vfio-pci` driver to the devices

```
echo "blacklist radeon" >> /etc/modprobe.d/blacklist.conf
echo "blacklist nouveau" >> /etc/modprobe.d/blacklist.conf
echo "blacklist nvidia" >> /etc/modprobe.d/blacklist.conf
```

B. Enable the IOMMU for systemd-boot (Proxmox on UEFI)

Get *VendorID* and *DeviceID* of the GPU

```
lspci -nn | grep -i nvidia
```

Sample output

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP106 [GeForc
01:00.1 Audio device [0403]: NVIDIA Corporation GP106 High Definition Audi
```

=> For the first one: *PCI ID*: `01:00.0` , *VendorID*: `10de` , *DeviceID*: `1c03`

Add kernel parameters

Add the following parameters into `/etc/kernel/cmdline`

```
intel_iommu=on vfio-pci.ids=<VendorID>:<DeviceID>,<VendorID>:<DeviceID> di
```

For example `/etc/kernel/cmdline` becomes

```
root=ZFS=rpool/ROOT/pve-1 boot=zfs intel_iommu=on vfio-pci.ids=10de:1c03,1
```

Update and reboot

```
pve-efiboot-tool
reboot
```

Check if kernel parameters are loaded.

C. IOMMU Interrupt Remapping

It will not be possible to use PCI passthrough **without interrupt remapping**.

To identify if your system has support for interrupt remapping

```
dmesg | grep 'remapping'
```

Sample output

```
[    0.190148] DMAR-IR: Queued invalidation will be enabled to support x2a
[    0.191599] DMAR-IR: Enabled IRQ remapping in x2apic mode
```

If you see one of the following lines, then remapping **is supported**.

- AMD-Vi: `Interrupt remapping enabled`
- DMAR-IR: `Enabled IRQ remapping in x2apic mode" ('x2apic' can be different on old CPUs, but should still work)`

Allow Unsafe Interrupts (if interrupt remapping is not supported)

If your system **doesn't support** interrupt remapping, you can allow **unsafe interrupts** (not tested by me)

```
echo "options vfio_iommu_type1 allow_unsafe_interrupts=1" > /etc/modprobe.
update-initramfs -u
reboot
```

D. Verification

Kernel Parameters Loaded

```
cat /proc/cmdline
```

IOMMU Working

```
dmesg | grep -E "DMAR|IOMMU"
```

Sample output

```
[    0.009442] ACPI: DMAR 0x000000007936D000 0000A8 (v01 INTEL  EDK2     0
[    0.110221] DMAR: IOMMU enabled
[    0.190123] DMAR: Host address width 39
[    0.190125] DMAR: DRHD base: 0x000000fed90000 flags: 0x0
[    0.190131] DMAR: dmar0: reg_base_addr fed90000 ver 1:0 cap 1c0000c4066
[    0.190133] DMAR: DRHD base: 0x000000fed91000 flags: 0x1
[    0.190137] DMAR: dmar1: reg_base_addr fed91000 ver 1:0 cap d2008c40660
[    0.190140] DMAR: RMRR base: 0x00000079945000 end: 0x00000079b8efff
[    0.190142] DMAR: RMRR base: 0x0000007b000000 end: 0x0000007f7fffff
[    0.190144] DMAR-IR: IOAPIC id 2 under DRHD base  0xfed91000 IOMMU 1
[    0.190146] DMAR-IR: HPET id 0 under DRHD base 0xfed91000
[    0.190148] DMAR-IR: Queued invalidation will be enabled to support x2a
[    0.191599] DMAR-IR: Enabled IRQ remapping in x2apic mode
[    0.942691] DMAR: No ATSR found
[    0.942753] DMAR: dmar0: Using Queued invalidation
[    0.942757] DMAR: dmar1: Using Queued invalidation
[    0.951433] DMAR: Intel(R) Virtualization Technology for Directed I/O
```

VFIO Working

```
dmesg | grep -i vfio
```

Should see messages from `vfio_pci` driver.

Sample output

```
[    0.000000] Command line: initrd=\EFI\proxmox\5.4.34-1-pve\initrd.img-5
[    0.110162] Kernel command line: initrd=\EFI\proxmox\5.4.34-1-pve\initr
[    0.987220] VFIO - User Level meta-driver version: 0.3
[    0.987271] vfio-pci 0000:01:00.0: vgaarb: changed VGA decodes: olddeco
[    1.006157] vfio_pci: add [10de:1c03[ffffffff:ffffffff]] class 0x000000
[    1.026154] vfio_pci: add [10de:10f1[ffffffff:ffffffff]] class 0x000000
[    5.320737] vfio-pci 0000:01:00.0: vgaarb: changed VGA decodes: olddeco
[   36.517767] vfio-pci 0000:01:00.0: vfio_ecap_init: hiding ecap 0x19@0x9
[   36.520535] vfio-pci 0000:01:00.0: Invalid PCI ROM header signature: ex
[   36.538147] vfio-pci 0000:01:00.1: enabling device (0000 -> 0002)
[   39.276682] vfio-pci 0000:01:00.0: Invalid PCI ROM header signature: ex
```

VF-PCI Driver Loaded

```
lspci -nnk
```

Kernel driver in use should be: `vfio-pci`

Sample output

```
...
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation GP106 [GeForc
        Subsystem: Micro-Star International Co., Ltd. [MSI] GP106 [GeForce
        Kernel driver in use: vfio-pci
        Kernel modules: nvidiafb, nouveau
01:00.1 Audio device [0403]: NVIDIA Corporation GP106 High Definition Audi
        Subsystem: Micro-Star International Co., Ltd. [MSI] GP106 High Def
        Kernel driver in use: vfio-pci
        Kernel modules: snd_hda_intel
...
```

IOMMU Groups Isolation

For working PCI passthrough, you need a **dedicated IOMMU group** for all PCI devices you want to assign to a VM.

```
find /sys/kernel/iommu_groups/ -type l
```

Sample output

```
/sys/kernel/iommu_groups/7/devices/0000:00:1c.0
/sys/kernel/iommu_groups/7/devices/0000:04:00.3
/sys/kernel/iommu_groups/7/devices/0000:04:00.1
/sys/kernel/iommu_groups/7/devices/0000:04:00.2
/sys/kernel/iommu_groups/7/devices/0000:00:1c.4
/sys/kernel/iommu_groups/7/devices/0000:04:00.0
/sys/kernel/iommu_groups/5/devices/0000:00:17.0
/sys/kernel/iommu_groups/3/devices/0000:00:14.0
/sys/kernel/iommu_groups/1/devices/0000:00:01.0
/sys/kernel/iommu_groups/1/devices/0000:01:00.0
/sys/kernel/iommu_groups/1/devices/0000:01:00.1
/sys/kernel/iommu_groups/8/devices/0000:05:00.0
/sys/kernel/iommu_groups/8/devices/0000:00:1d.0
/sys/kernel/iommu_groups/6/devices/0000:00:1b.0
/sys/kernel/iommu_groups/4/devices/0000:00:16.0
/sys/kernel/iommu_groups/2/devices/0000:00:02.0
/sys/kernel/iommu_groups/0/devices/0000:00:00.0
/sys/kernel/iommu_groups/9/devices/0000:00:1f.2
/sys/kernel/iommu_groups/9/devices/0000:00:1f.0
/sys/kernel/iommu_groups/9/devices/0000:00:1f.3
/sys/kernel/iommu_groups/9/devices/0000:00:1f.6
/sys/kernel/iommu_groups/9/devices/0000:00:1f.4
```

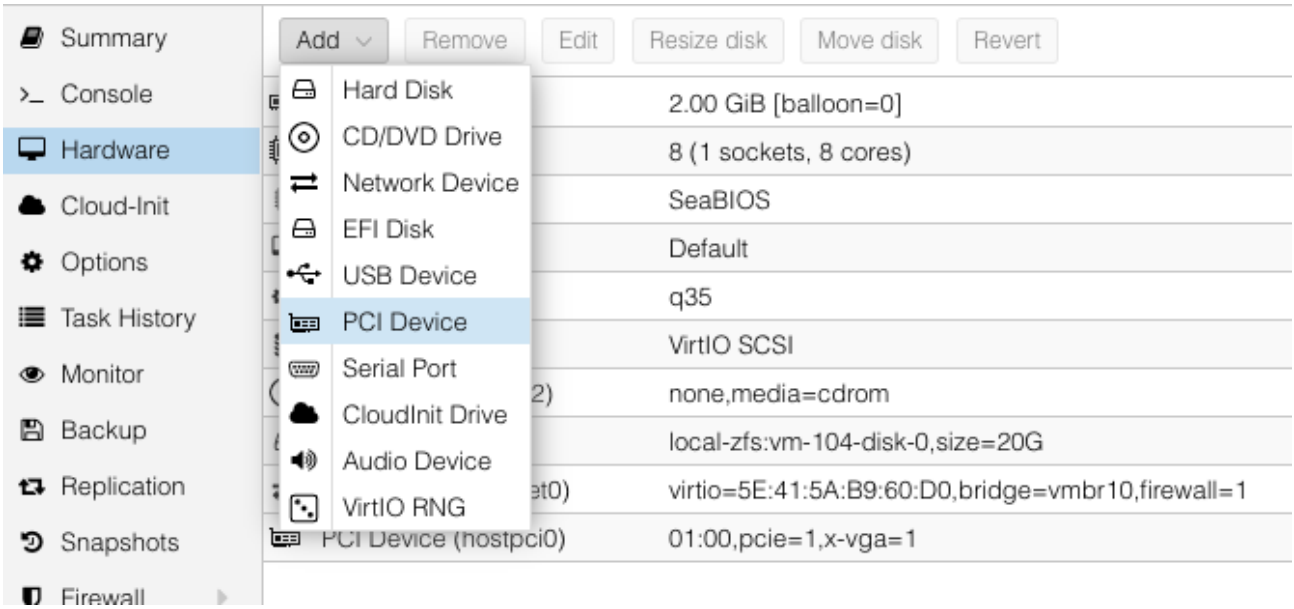
Guest VM Config

Guest OS: Ubuntu 18.04.4 Server

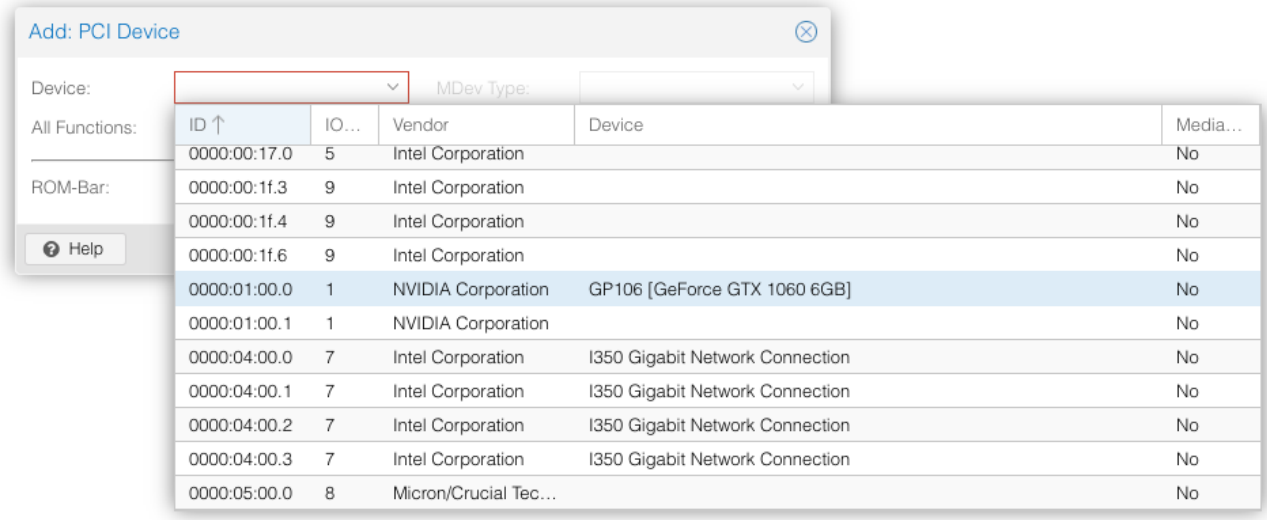
A. Add PCI Device to VM

Only user **root** can add PCI device to guest VM

```
Hardware => Add => PCI Device
```

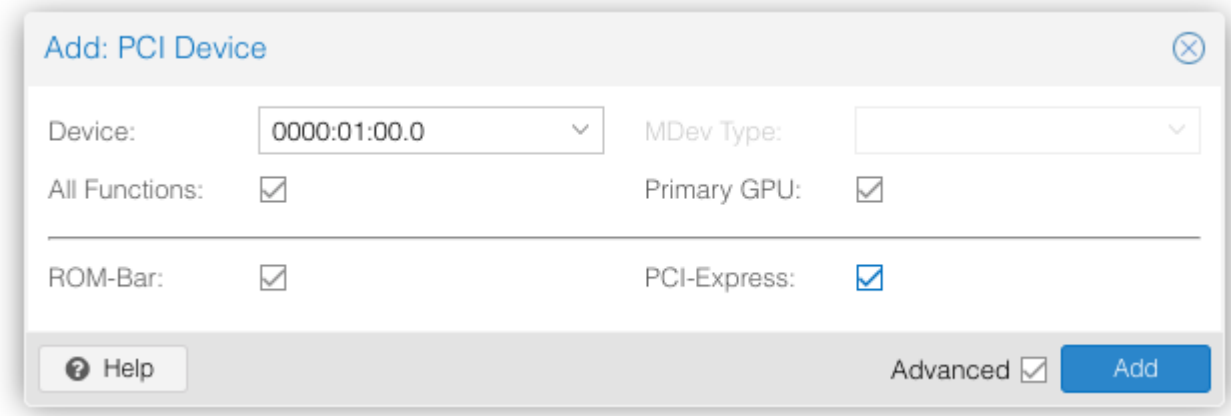


If the steps above are successfully done, should see the GPU here



Check all options

```
[v] All Functions
[v] Primary GPU
[v] ROM-Bar
[v] PCI-Express
```



B. Other VM Hardware Configs

Choose either option 1 or option 2 to configure.

Warning: lost of the web GUI terminal access

> Once the *Primary GPU* (`x-vga=1`) option is set, the VNC consle on Web GUI will **NOT** able to connect to the VM, so make sure you are able to access the VM by **SSH**, and the network setting in the VM will not be changed after reboot.

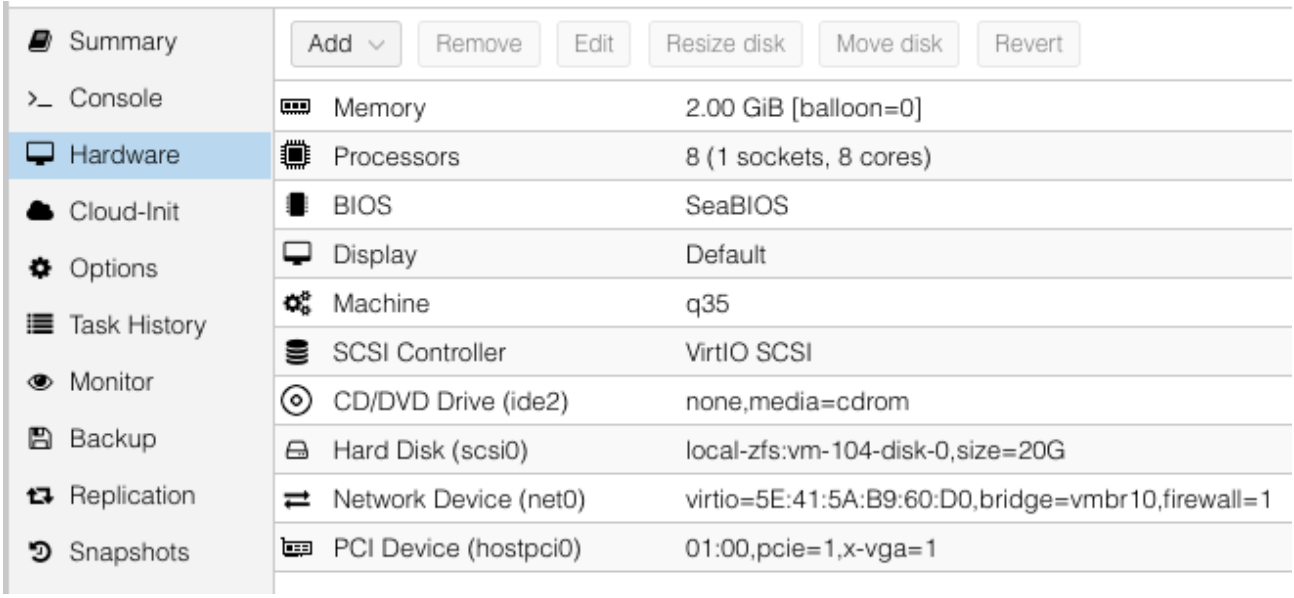
Option 1. Edit the VM Config

Edit `/etc/pve/qemu-server/<VMID>.conf`

```
bios: seabios
machine: q35
hostpci0: 01:00,pcie=1,x-vga=1
```

Option 2. By the Web GUI

Set *BIOS*, *Machine*, *PCI Device (hostpci0)* by the Web GUI



Guest OS Config

A. Prerequisities

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt update
sudo apt install ubuntu-drivers-common
```

B. Install NVIDIA Driver

Check the latest available driver for the GPU

```
ubuntu-drivers devices
```

Sample Output

```
== /sys/devices/pci0000:00/0000:00:1c.0/0000:01:00.0 ==
modalias : pci:v000010DEd00001C03sv00001462sd00003283bc03sc00i00
vendor   : NVIDIA Corporation
model    : GP106 [GeForce GTX 1060 6GB]
driver   : nvidia-driver-415 - third-party free
driver   : nvidia-driver-435 - distro non-free
driver   : nvidia-driver-440 - distro non-free
driver   : nvidia-driver-440-server - distro non-free
driver   : nvidia-driver-390 - distro non-free
driver   : nvidia-driver-410 - third-party free
driver   : nvidia-driver-450 - third-party free recommended
driver   : nvidia-driver-418-server - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
```

=> The output shows that we can install `nvidia-driver-440-server`

```
sudo apt install nvidia-driver-440-server nvidia-utils-440-server
sudo reboot
```

Verification

```
lsmod | grep nvidia
```

C. GRUB Options

Warning: lost of access to the VNC terminal

> There will be no output to the VNC terminal after these options are set, make the ssh is ready.

Add the following options to `GRUB_CMDLINE_LINUX_DEFAULT` in `/etc/default/grub`

```
video=vesafb:off,efifb:off
```

Run

```
sudo update-grub
sudo reboot
```

D. Verification

```
sudo nvidia-smi
```

Sample output

```
Sat Aug  1 19:45:48 2020
+-----------------------------------------------------------------------------
| NVIDIA-SMI 440.95.01    Driver Version: 440.95.01    CUDA Version: 10.2
|-------------------------------+----------------------+----------------------
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr.
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute
|===============================+======================+======================
|   0  GeForce GTX 106...  Off  | 00000000:01:00.0 Off |
|  0%   45C    P5    14W / 140W |      0MiB /  6078MiB |      0%      Defa
+-------------------------------+----------------------+----------------------

+-----------------------------------------------------------------------------
| Processes:                                                       GPU Mem
|  GPU       PID   Type   Process name                             Usage
|=============================================================================
|  No running processes found
+-----------------------------------------------------------------------------
```

=> The GPU information is successfully retrieved without any error.

References

BIOS

- https://us.informatiweb.net/tutorials/it/bios/enable-iommu-or-vt-d-in-your-bios.html (https://us.informatiweb.net/tutorials/it/bios/enable-iommu-or-vt-d-in-your-bios.html)

PVE

- https://pve.proxmox.com/wiki/Pci_passthrough (https://pve.proxmox.com/wiki/Pci_passthrough)
- https://mathiashueber.com/pci-passthrough-ubuntu-2004-virtual-machine/ (https://mathiashueber.com/pci-passthrough-ubuntu-2004-virtual-machine/) https://stackoverflow.com/questions/48199261/proc-cmdline-does-not-updated-with-update-grub (https://stackoverflow.com/questions/48199261/proc-cmdline-does-not-updated-with-update-grub)
- https://pve.proxmox.com/wiki/PCI(e)_Passthrough (https://pve.proxmox.com/wiki/PCI(e)_Passthrough) (outdated)

Guest OS

- https://gitpress.io/@chchang/install-nvidia-driver-cuda-pgstrom-in-ubuntu-1804 (https://gitpress.io/@chchang/install-nvidia-driver-cuda-pgstrom-in-ubuntu-1804)
- https://www.facebook.com/groups/pve.tw/permalink/1556562391178983/ (https://www.facebook.com/groups/pve.tw/permalink/1556562391178983/)