Table of contents

Getting Started

Here is what I've learned regarding Virtualization in Arch Linux (btw)

At the end of this guide we will have a nice Windows 11 VM with pass-through GPU NVIDIA, and seamless capture of mouse and keyboard.

We will use **KVM / QEMU** with **libvirt** and our GUI will be **virt-manager**

Arch Linux uses PipeWire for audio (this we'll use for sending guest audio back to host, without the needs of a virtual monitor attached)

I won't use **Spice** or **Looking Glass**, therefore you will need to use a monitor hooked into your PCI GPU.

Before you start

List of the prerequisites that are required or recommended.

Make sure that:

- First: Your CPU and your MB Must be compatible with Virtualization, IOMMU.
- Second: You have at least 2 GPUS, internal GPU from an intel/amd CPU chip (APU) works.

We are going to also follow the reference from this wiki Arch Wiki (https://wiki.archlinux.org/title/PCI_passthrough_via_OVMF)

1. Setting up BIOS

First of all you need to go to your BIOS settings and activate:

1. **Virtualization Extensions**

   Intel VT-x (Intel Virtualization Technology for x86)

   - What: Allows a CPU to act as if you have multiple independent computers, called virtual machines (VMs).
   - How: In BIOS/UEFI, look for options like "Intel Virtualization Technology," "VT-x," or "Intel VT-x," and enable them.

AMD-V (AMD Virtualization)

- What: Similar to Intel VT-x, it allows hardware-assisted virtualization on AMD processors.
- How: In BIOS/UEFI, find settings like "SVM Mode" or "Secure Virtual Machine" and enable them.

2. **IOMMU (Input-Output Memory Management Unit)**

- What: IOMMU maps physical memory to virtual memory, which is critical for device passthrough (e.g., PCIe devices) in virtualized environments. It ensures that VMs can securely and efficiently use hardware devices.

Intel: Known as VT-d (Intel Virtualization Technology for Directed I/O).

- How: Enable "VT-d" or "Intel VT for Directed I/O" in the BIOS/UEFI settings.

AMD: Often referred to simply as "IOMMU."

- How: Enable "IOMMU" or "AMD-Vi" in the BIOS/UEFI.

2. Check IOMMU

Ensure Your Kernel Supports IOMMU and VFIO:

```
LC_ALL=C.UTF-8 lscpu | grep Virtualization
```

See kernel support:

```
zgrep CONFIG_KVM= /proc/config.gz
lsmod | grep kvm
lsmod | grep vfio
lsmod | grep virtio
```

3. Configure Grub

We are going to configure GRUB first.

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

Add intel or amd iommu to on:

```
GRUB_CMDLINE_LINUX_DEFAULT=" ... intel_iommu=on ..."
```

Then update grub:

```
sudo grub-mkconfig -o /boot/grub/grub.cfg
```

4. Load Missing Modules

In case something was not correctly loaded when checking IOMMU, we are going to load the missing modules.

Create a file /etc/modules-load.d/vfio.conf

And write the modules

```
sudo nano /etc/modules-load.d/vfio.conf

vfio
vfio_iommu_type1
vfio_pci
```

5. Isolate the GPU

We need to find the IDs for our GPU, since we are going to pass-through.

```
lspci -nn
```

We will find a lot of information, but we only need to pick both the NVIDIA Graphics and it's Audio.

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation AD103
[GeForce RTX 4080] [10de:2704] (rev a1)
```

```
01:00.1 Audio device [0403]: NVIDIA Corporation Device [10de:22bb]
(rev a1)
```

Now we create a file /etc/modprobe.d/vfio.conf

```
options vfio-pci ids=10de:2704,10de:22bb
softdep nvidia pre: vfio-pci
```

I use my ids, from both the VGA and the Audio device, split by comma.

The *softdep* line is to make sure we isolate the GPU before the driver.

6. After Reboot.

We can check finally if everything we did is correct.

Here is a nice script to check the IOMMU groups:

```
#!/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;
```

If we execute this we can find if our GPU is in an isolated group. Mine's in group 13.

```
IOMMU Group 13:
    01:00.0 VGA compatible controller [0300]: NVIDIA Corporation
AD103 [GeForce RTX 4080] [10de:2704] (rev a1)
    01:00.1 Audio device [0403]: NVIDIA Corporation Device
[10de:22bb] (rev a1)
```

Now we can check which driver is taking our NVIDIA.

```
lspci -nnk
```

We should see that the driver in use is vfio-pci:

```
01:00.0 VGA compatible controller [0300]: NVIDIA Corporation AD103
[GeForce RTX 4080] [10de:2704] (rev a1)
    Subsystem: Micro-Star International Co., Ltd. [MSI] Device
[1462:5110]
    Kernel driver in use: vfio-pci
    Kernel modules: nouveau, nvidia_drm, nvidia
01:00.1 Audio device [0403]: NVIDIA Corporation Device [10de:22bb]
(rev a1)
    Subsystem: Micro-Star International Co., Ltd. [MSI] Device
[1462:5110]
    Kernel driver in use: vfio-pci
    Kernel modules: snd_hda_intel
```

This is cool, we can pass this GPU without issues!

You can continue next to setting up the virtual machine!

Setting up virtual machine

We need to install some packages:

```
sudo pacman -S qemu-desktop dnsmasq virt-manager ebtables libvirt
edk2-ovmf swtpm
```

We will need to activate the default NAT network:

```
virsh net-autostart default
virsh net-start default
```

The process of setting up a virtual machine using virt-manager is mostly self-explanatory, as most of the process comes with fairly comprehensive on-screen instructions.

However, you should pay special attention to the following steps:

- When the virtual machine creation wizard asks you to name your virtual machine (**final step before clicking "Finish"**), check the **"Customize before install" checkbox**.
- Select the BIOS and search for the UEFI with secureboot for our x64 platform.
- In the "CPUs" section, adjust your socket, cores and threads per core, make sure to verify you are sending the correct setup.

Since we are installing Windows 11, we need also a TPM2. Since we installed *swtpm*, we can "Add Hardware" and we should see a TPM2 Emulator.

(Check also if you want to use virtio for your main disk now)

Then we are ready to install the OS as normal.

VM Configuration

Now we are going to configure the VM.

First of all, shut down your VM!

We need to set up Virt-Manager under Settings or Preferences, to allow us to edit XML.

Mouse & Keyboard (evdev)

We are going to tackle the mouse and keyboard. Right now we have Spice and a virtual monitor to us, which captures the mouse and keyboard, however is best to send them using evdev.

Firs we need to get the devices ID of our mouse and keyboard:

```
ls -l /dev/input/by-id/
```

You will get something like this:

```
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
Corsair_CORSAIR_HS80_MAX_WIRELESS_Gaming_Receiver_592466F35C4EC86F
-event-if03 -> ../event24
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
Corsair_LCD_Cap_for_Elite_Capellix_coolers_1416221080004-event-
if00 -> ../event22
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
Microsoft_Controller_303236333031313538353232303 3-event-joystick -
> ../event26
lrwxrwxrwx 1 root root  6 Jul 24 19:57 usb-
Microsoft_Controller_303236333031313538353232303 3-joystick ->
../js0
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
monsgeek_MonsGeek_Keyboard-event-if02 -> ../event15
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
monsgeek_MonsGeek_Keyboard-event-kbd -> ../event14
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
monsgeek_MonsGeek_Keyboard-if02-event-kbd -> ../event17
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
```

```
Razer_Razer_Viper_Ultimate_000000000000-event-if01 -> ../event20
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
Razer_Razer_Viper_Ultimate_000000000000-event-mouse -> ../event18
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
Razer_Razer_Viper_Ultimate_000000000000-if01-event-kbd ->
../event19
lrwxrwxrwx 1 root root 10 Jul 24 19:57 usb-
Razer_Razer_Viper_Ultimate_000000000000-if02-event-kbd ->
../event21
lrwxrwxrwx 1 root root  9 Jul 24 19:57 usb-
Razer_Razer_Viper_Ultimate_000000000000-mouse -> ../mouse0
```

Im going to use my MonsGeek keyboard and my Razer Viper mouse. For that we need to look at the event name of each one. In my case they are:

```
usb-Razer_Razer_Viper_Ultimate_000000000000-event-mouse
usb-monsgeek_MonsGeek_Keyboard-event-kbd
```

IMPORTANT

Some devices will have other event like my keyboard has: **usb-monsgeek_MonsGeek_Keyboard-if02-event-kbd**

Sometimes when adding these devices, and we want to get back the mouse and keyboard to the host, it will only pass the keyboard and not the mouse.

For that, we need to experiment, usually it is the if02 event of the keyboard. In my case, I need it, so I will use it from now on.

Configure the VM XML file.

Now that we have our devices, we will edit the VM using xml. (We can use a command or the virt-manager editor)

```
sudo EDITOR=nano virsh edit nameofyourvm
```

We will add this configuration:

```
...
  <devices>
```

```
    ...
      <input type="evdev">
          <source dev="/dev/input/by-id/usb-
monsgeek_MonsGeek_Keyboard-if02-event-kbd" grab="all"
grabToggle="ctrl-ctrl" repeat="on"/>
      </input>
      <input type="evdev">
          <source dev="/dev/input/by-id/usb-
monsgeek_MonsGeek_Keyboard-event-kbd"/>
      </input>
      <input type="evdev">
          <source dev="/dev/input/by-id/usb-
Razer_Razer_Viper_Ultimate_000000000000-event-mouse"/>
      </input>
    ...
  </devices>
```

You can see my if02-event-kbd has the options grab grabToggle and repeat, this is the device which recognized better in my machine. Instead of the regular -event-kbd. For you it might work with the regular one or you might have to use one of the if02.

In either case, you need to also pass the default -event-kbd if you needed the if02 as you see in my example.

Check

If we boot the machine now, the VM will take instantly control over our mouse and keyboard, if you press `CTRL + CTRL`, both CTRL buttons on your keyboard, the mouse and keyboard should respond back in the host. If that is not the case, please check the step before.

GPU pass-through

This is actually an easy one, we just isolated the graphics card, so we can just go over the GUI (virt-manager), click on 'Add Hardware' and select "PCI Host Device", we should find our NVIDIA VGA and its Audio Device, we need to add both.

Now the VM will have direct access to it.

Disable Virtual Monitor

In my setup I use a DisplayPort cable connected to my monitor second input, so I do not need to see or have a screen sharing of the machine.

Therefore, I will disable spice, and the monitor (virtual one). For that, we need to remove these entries from the XML all at once.

```
...
<audio id="1" type="spice"/>
...
<graphics type="spice" autoport="yes">
    <listen type="address"/>
    <image compression="off"/>
</graphics>
...
<channel type="spicevmc">
    <target type="virtio" name="com.redhat.spice.0"/>
    <address type="virtio-serial" controller="0" bus="0"
port="1"/>
</channel>
...
<redirdev bus="usb" type="spicevmc">
    <address type="usb" bus="0" port="1"/>
</redirdev>
...
```

Save the file and start your machine again, nothing will display but your external monitor should show you still the DisplayPort signal, with only that monitor. (Physical)

Include Virtio Disks

If you have installed Virtio drivers from redhat / fedora binaries to your window's machine. Check Wiki Virtio (https://wiki.archlinux.org/title/QEMU#Installing_virtio_drivers) You can hook up any other internal drivers you have to that machine like this:

```
<disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='writeback' io='threads'
```

```
discard='unmap'/>
    <source dev='/dev/disk/by-id/ata-
Samsung_SSD_870_EVO_4TB_S758NX0W502467Z'/>
    <target dev='sda' bus='scsi'/>
</disk>
<controller type='scsi' index='0' model='virtio-scsi'>
    <driver iothread='1' queues='8'/>
</controller>
```

If you get an error about io threads add also this line:

```
<iothreads>1</iothreads>
```

Get Audio from Guest to your Host

If we want to hear the audio from the guest to the host, we can use PipeWire.

We can add this to our XML config:

```
<audio id="1" type="pipewire" runtimeDir="/run/user/1000">
    <input name="qemuinput"/>
    <output name="qemuoutput"/>
</audio>
```

Of course make sure you are user 1000 or set the user yourself.