

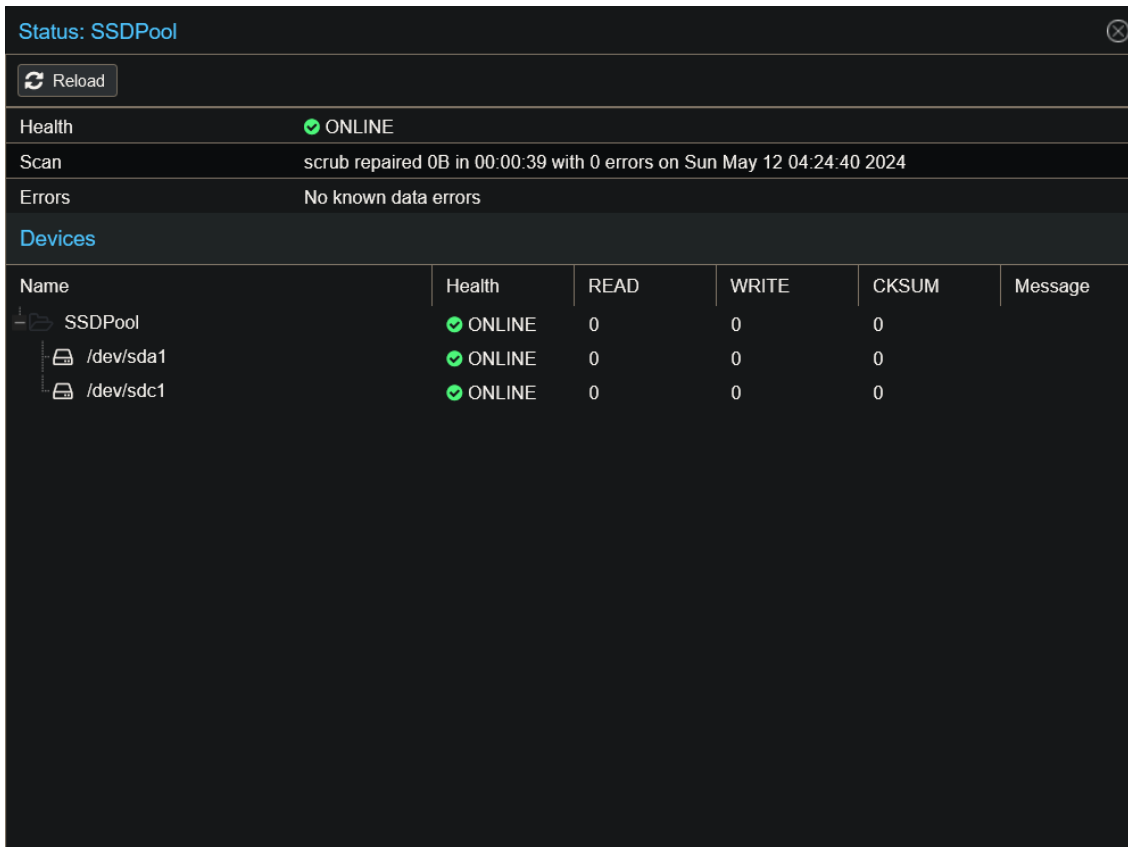
09 - Proxmox

All things about Proxmox

- [Backing up Proxmox VMs if you have a storage pool already](#)
- [Project: Proxmox Standalone GPU Passthrough Server Build](#)
 - [Project: Proxmox Standalone GPU Passthrough Server Build](#)
 - [Installing Proxmox VE 8.2 on a Standalone GPU Passthrough Server](#)
 - [Installing Pop!_OS \(NVIDIA Edition\) in Proxmox](#)
 - [Setting Up SSH Access in Pop!_OS \(Proxmox VM\)](#)
 - [NVIDIA GPU Passthrough in Pop!_OS \(Proxmox VE 8.2.7\)](#)

Backing up Proxmox VMs if you have a storage pool already

I have a storage pool made from 2 500GB SSDs in Proxmox.
I have it made in ZFS.



The screenshot shows the Proxmox storage pool status for 'SSDPool'. The status is 'ONLINE'. A scan was performed on Sun May 12 04:24:40 2024, scrubbing 0B with 0 errors. There are no known data errors. The devices section shows the SSDPool and its components: /dev/sda1 and /dev/sdc1, all of which are ONLINE with 0 reads and 0 writes.

Name	Health	READ	WRITE	CKSUM	Message
SSDPool	ONLINE	0	0	0	
/dev/sda1	ONLINE	0	0	0	
/dev/sdc1	ONLINE	0	0	0	

Enter the Proxmox shell:

```
apt-update
```

Then: **Replace the SSDPool with the name of your pool**

My pool is in ZFS so I can make the folder a different way than mkdir:

```
zfs create SSDPool/backups
```

Go to Datacenter > Backup > Add

Add: Directory

General | Backup Retention

ID: Nodes:

Directory: Enable:

Content: Shared:

Preallocation:

Advanced

It should now be in your list of storage in Proxmox:

PROXMOX Virtual Environment 8.1.3

Server View | Datacenter

Storage List:

ID	Type	Content ↑	Path/Target	S...	E...	Bandwidt...
SSD_Bac...	Directory	VZDump backup ...	/SSDPool...	No	Yes	
local	Directory	VZDump backup ...	/var/lib/vz	No	Yes	
SSDPool	ZFS	Disk image, Cont...		No	Yes	
local-lvm	LVM-T...	Disk image, Cont...		No	Yes	

Now move on to creating the backup job.

Go to Datacenter > Backup > Add Job

Storage: Select your storage you made

Schedule: Mine is everyday at 21:00

Selection Mode: Include Selected VM's

I have this backing up BOOKSTACK and WIN-SRV-22:

Edit: Backup Job ✕

General | Retention | Note Template

Node: proxmox ✕ | Notification mode: Default (Auto) ▼

Storage: SSD_Backups_Dir ▼ | Send email: Always ▼

Schedule: 21:00 ▼ | Send email to:

Selection mode: Include selected VMs ▼ | Compression: ZSTD (fast and good) ▼

Mode: Snapshot ▼

Enable:

Job Comment: Backing up WIN-SRV-22 and BOOKSTACK

<input type="checkbox"/>	ID ↑	Node	Status	Name	Type
<input checked="" type="checkbox"/>	100	proxmox	running	WIN-SRV-22	Virtual Machine
<input type="checkbox"/>	101	proxmox	stopped	DOCKER-UBUNTU	Virtual Machine
<input checked="" type="checkbox"/>	102	proxmox	running	BOOKSTACK	Virtual Machine
<input type="checkbox"/>	103	proxmox	stopped	Netboot	Virtual Machine

Repeat missed:

? Help Advanced OK Reset

Hit OK.

Run the backup now to test:

Datacenter ? Help

+ Add - Remove E Edit J Job Detail R **Run now** i Show: Guests Without Backup Job S Schedule Simulator

Enabled	Node	Schedule	Next Run	Storage	Comment	Retention ↑
<input checked="" type="checkbox"/>	proxmox	21:00	2024-06-06 21:00:00	SSD_Back...	Backing up WIN-SRV-22 and BOOKSTACK	Fallback from stora

- Search
- Summary
- Notes
- Cluster
 - Ceph
- Options
- Storage
- Backup**
- Replication
- Permissions
 - Users
 - API Tokens
 - Two Factor
 - Groups
 - Pools
 - Roles
 - Roles

You will be able to see it in the Task Lists of Proxmox and Double Clicking will show you the current operations:

Proxmox Virtual Environment 8.1.3

Server View | Datacenter

Search

Documentation | Create VM | Create CT | root@pam

Tasks | Cluster log

Enabled	Node	Schedule	Next Run	Storage	Comment	Retention ↑
✓	proxmox	21.00	2024-06-06 21:00:00	SSD_Back...	Backing up WIN-SRV-22 and BOOKSTACK	Fallback from stora

Start Time ↓	End Time	Node	User name	Description	Status
Jun 06 12:19:41		proxmox	root@pam	Backup Job	
Jun 06 11:57:30	Jun 06 12:07:40	proxmox	root@pam	Shell	OK
Jun 06 11:51:05	Jun 06 11:53:45	proxmox	root@pam	Shell	OK
Jun 06 10:37:14	Jun 06 10:37:25	proxmox	root@pam	VM/CT 102 - Console	OK

Double Click it:

Task viewer: Backup Job

Output | Status

Stop | Download

```

INFO: include disk 'tpmstate0' 'local-lvm:vm-100-disk-2' 4M
INFO: backup mode: snapshot
INFO: ionice priority: 7
INFO: creating v2dump archive '/SSDPool/backups/dump/v2dump-qemu-100-2024_06_06-12_19_41.vma.zst'
INFO: attaching TPM drive to QEMU for backup
INFO: started backup task 'c41f3e18-cdf6-4f73-a0d5-b8bd8cd48085'
INFO: resuming VM again
INFO: 0% (560.3 MiB of 80.0 GiB) in 3s, read: 186.8 MiB/s, write: 148.1 MiB/s
INFO: 1% (1.1 GiB of 80.0 GiB) in 6s, read: 180.4 MiB/s, write: 178.1 MiB/s
INFO: 2% (1.7 GiB of 80.0 GiB) in 10s, read: 163.8 MiB/s, write: 161.5 MiB/s
INFO: 3% (2.4 GiB of 80.0 GiB) in 15s, read: 150.4 MiB/s, write: 150.1 MiB/s
INFO: 4% (3.3 GiB of 80.0 GiB) in 21s, read: 147.6 MiB/s, write: 146.3 MiB/s
INFO: 5% (4.0 GiB of 80.0 GiB) in 25s, read: 176.4 MiB/s, write: 175.1 MiB/s
INFO: 6% (5.0 GiB of 80.0 GiB) in 31s, read: 165.3 MiB/s, write: 165.0 MiB/s
INFO: 7% (5.7 GiB of 80.0 GiB) in 36s, read: 156.1 MiB/s, write: 154.5 MiB/s
INFO: 8% (6.5 GiB of 80.0 GiB) in 41s, read: 154.3 MiB/s, write: 152.3 MiB/s
INFO: 9% (7.3 GiB of 80.0 GiB) in 46s, read: 159.6 MiB/s, write: 157.8 MiB/s
INFO: 10% (8.1 GiB of 80.0 GiB) in 51s, read: 163.4 MiB/s, write: 161.8 MiB/s
INFO: 11% (8.8 GiB of 80.0 GiB) in 56s, read: 158.5 MiB/s, write: 157.9 MiB/s
INFO: 12% (9.7 GiB of 80.0 GiB) in 1m 2s, read: 148.6 MiB/s, write: 148.2 MiB/s
INFO: 13% (10.6 GiB of 80.0 GiB) in 1m 7s, read: 173.3 MiB/s, write: 172.6 MiB/s
INFO: 14% (11.3 GiB of 80.0 GiB) in 1m 11s, read: 184.9 MiB/s, write: 182.0 MiB/s
INFO: 15% (12.1 GiB of 80.0 GiB) in 1m 16s, read: 168.9 MiB/s, write: 166.2 MiB/s
INFO: 16% (12.9 GiB of 80.0 GiB) in 1m 20s, read: 200.0 MiB/s, write: 198.8 MiB/s

```

Now you can have a backup of selected VM's

You can find the backups in your SSD pool > Backups:

- Server View
- Datacenter
 - proxmox
 - 100 (WIN-SRV-22)
 - 101 (DOCKER-UBUNTU)
 - 102 (BOOKSTACK)
 - 103 (Netboot)
 - localnetwork (proxmox)
 - SSDPool (proxmox)
 - SSD_Backups_Dir (proxmox)
 - local (proxmox)
 - local-lvm (proxmox)

Storage 'SSD_Backups_Dir' on node 'proxmox' Help

Summary Restore Show Configuration Edit Notes Change Protection Prune group Remove Search: Name, Format, Notes

Backups

Permissions

Name	Notes	Date ↓	Format	Size
vzdump-qemu-102-2024_06_06-12_22_41.vma.zst	BOOKSTACK	2024-06-06 12:22:41	vma.zst	15.38 GB
vzdump-qemu-100-2024_06_06-12_19_41.vma.zst	WIN-SRV-22	2024-06-06 12:19:41	vma.zst	12.66 GB

Project: Proxmox Standalone GPU Passthrough Server Build

This Chapter is for a proxmox build and passing a GPU into one of the virtual machines.

Project: Proxmox Standalone GPU Passthrough Server Build

Date: May 31st 2025

Category: Virtualization / Homelab Build

Hardware Overview

Component	Model / Spec
Motherboard	Gigabyte B550 AORUS Elite AX V2
CPU	AMD Ryzen 7 3700X (8-core / 16-thread)
GPU	NVIDIA RTX 2070 Super
RAM	32GB DDR4 3200 MHz
Drive 1 (OS)	Kioxia 512GB NVMe Gen4
Drive 2 (VMs)	Inland 512GB NVMe Gen3
Host OS	Proxmox VE 8.2.7
Primary VM	Pop!_OS 22.04 LTS (NVIDIA ISO)

BIOS Setup for Proxmox + GPU Passthrough

Motherboard: Gigabyte B550 AORUS Elite AX V2

1. Enter BIOS

- Press `DEL` repeatedly during boot to enter BIOS.
-

2. Load Optimized Defaults (Recommended)

- Go to the **Save & Exit** tab.
 - Choose **Load Optimized Defaults**
 - Press `F10` to reboot and re-enter BIOS.
-

3. Enable Virtualization (SVM Mode)

- Navigate to: `Tweaker → Advanced CPU Settings`
 - Set **SVM Mode** → `Enabled`
-

4. Enable IOMMU

- Go to: `Settings → IO Ports`
 - Set **IOMMU** → `Enabled`
-

5. Enable Above 4G Decoding

- Also under: `Settings → IO Ports`
 - Set **Above 4G Decoding** → `Enabled`
-

6. Set Initial Display Output

- Go to: `Settings → IO Ports`
 - Set **Initial Display Output** → `PCIe 1 Slot`
-

7. Disable CSM for UEFI Boot

- Go to: `Boot`
 - Set **CSM Support** → `Disabled`
-

8. Resizable BAR Support (Optional)

- `Settings → IO Ports`
 - Set **Re-Size BAR Support** → `Disabled` (*recommended for stability*)
-

9. Fan Control (Optional)

- `Smart Fan 5 Settings`
 - Set profiles for optimal cooling: Silent, Normal, Full-Speed
-

10. Save and Exit

- Press `F10`, confirm changes, and reboot.

Now its time to install proxmox on the new setup.

Installing Proxmox VE 8.2 on a Standalone GPU Passthrough Server

Date: June 1st, 2025

Category: Virtualization / Proxmox Deployment

Backlink: [Project: Proxmox Standalone GPU Passthrough Server Build](#)

Requirements

- Proxmox VE 8.2 ISO Installer
 - Bootable USB creation tool (Etcher or Rufus)
 - At least 1 NVMe SSD (recommended Gen4 for performance)
 - BIOS configured for UEFI boot, IOMMU, virtualization, and passthrough (see backlink for full BIOS setup)
-

Step 1: Download and Flash Proxmox ISO

1. Go to [Proxmox Downloads](#)
2. Download the latest ISO: `proxmox-ve_8.2-1.iso` (or newer)
3. Use **Balena Etcher** or **Rufus** to flash it to a USB drive
 - Select ISO
 - Select USB
 - Click **Flash**

Step 2: Install Proxmox VE

1. Boot from the USB using **UEFI Boot Mode**
2. Select **Install Proxmox VE** from the menu
3. Accept license agreement
4. Choose the **Gen4 NVMe** (Kioxia in this case) as the target disk
5. Configure:
 - Region & Timezone
 - Strong root password & email
 - Hostname (e.g. `proxmox-node2.local`)
 - Static IP address (e.g. `192.168.1.101`) or use DHCP for testing

Step 3: First Boot and Web GUI Access

- Remove the USB when prompted
- Log into the Proxmox web GUI:

`https://<your-proxmox-ip>:8006`

- Accept the self-signed certificate

Step 4: Secondary Drive (Optional)

If you have a second NVMe (like an Inland Gen3):

1. Go to `Datacenter > Disks`
2. Select `/dev/nvme1n1`
3. Wipe the disk
4. Initialize it with GPT
5. Create a new storage:
 - As **LVM-Thin** for VM disk storage
 - Or as **Directory** for ISOs/backups

Installing Pop!_OS (NVIDIA Edition) in Proxmox

Date: June 1st, 2025

Category: Virtualization / VM Guest OS Configuration

Backlink: [Installing Proxmox VE 8.2 on a Standalone GPU Passthrough Server](#)

Goal

Install **Pop!_OS 22.04 LTS (NVIDIA ISO)** as a Proxmox VM with a **passed-through RTX 2070 Super GPU**, ensuring full graphics acceleration and NVIDIA driver functionality.

Step 1: Download the Pop!_OS NVIDIA ISO

Download directly to Proxmox or to your workstation:

[Pop!_OS 22.04 LTS NVIDIA ISO](#)

Option 1: Upload via Proxmox GUI

- Go to `Datacenter > local (proxmox89) > ISO Images`
- Click **Upload** and choose the ISO file from your computer

Option 2: Download from URL

- In `ISO Images`, click ****Download from URL`**
- Paste the ISO link above and click **Query URL**
- Confirm filename and click **Download**

The ISO will appear in the list once it's downloaded.

Step 2: Create the Pop!_OS VM

General Tab

- Node: `proxmox89`
- VM ID: `100` (or next available)
- Name: `POP_OS_NVIDIA`
- Start at boot

OS Tab

- Use ISO: `pop-os_22.04_amd64_nvidia_53.iso`

System Tab

- BIOS: `OVMF (UEFI)`
- Machine: `q35`
- Add EFI Disk
- Pre-Enrolled Keys (disable Secure Boot)
- SCSI Controller: `VirtIO SCSI single`
- QEMU Agent
- Graphic Card: `Default` (or change to `None` later for passthrough)

Disks Tab

- Bus: `SCSI`
- Storage: `Inland512`
- Size: `200 GiB`
- IO thread

CPU Tab

- Sockets: 1
- Cores: 6-8
- Type: host

Memory Tab

- Size: 16384 MiB

Network Tab

- Bridge: vmbr0
- Model: VirtIO (paravirtualized)
- Firewall (optional)

Click **Finish** to create the VM.

Step 3: Attach GPU (Passthrough)

1. Stop the VM.
 2. Go to Hardware > Add > PCI Device
 3. Select both:
 - 01:00.0 NVIDIA VGA Controller
 - 01:00.1 NVIDIA HD Audio
 4. Enable:
 - All Functions
 - Primary GPU
 - ROM-Bar
-

Step 4: Boot and Install Pop!_OS

- Start the VM
- If passthrough is correct, output will show on monitor connected to GPU
- Use "Erase Disk and Install"

- Set username and password

Setting Up SSH Access in Pop!_OS (Proxmox VM)

Date: June 1st, 2025

Category: Remote Access / Virtual Machine Setup

Backlink: [Installing Pop!_OS \(NVIDIA Edition\) in Proxmox with GPU Passthrough](#)

Goal

Enable secure remote access to your Pop!_OS virtual machine via SSH.

Step 1: Install and Enable OpenSSH Server

Open a terminal in your Pop!_OS VM and run:

```
sudo apt update  
sudo apt install openssh-server -y
```

Then start and enable the service:

```
sudo systemctl enable ssh  
sudo systemctl start ssh
```

```
zippyb@pop-os:~$ sudo apt update -y
Hit:1 http://apt.pop-os.org/proprietary jammy InRelease
Hit:2 http://apt.pop-os.org/release jammy InRelease
Hit:3 http://apt.pop-os.org/ubuntu jammy InRelease
Hit:4 http://apt.pop-os.org/ubuntu jammy-security InRelease
Hit:5 http://apt.pop-os.org/ubuntu jammy-updates InRelease
Hit:6 http://apt.pop-os.org/ubuntu jammy-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
zippyb@pop-os:~$ sudo apt install openssh-server -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
```

Step 2: Verify SSH is Running

Check the status of the SSH server:

```
sudo systemctl status ssh
```

You should see:

- Active: active (running)

```
zippyb@pop-os:~$ sudo systemctl start ssh
zippyb@pop-os:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: e
   Active: active (running) since Sun 2025-06-01 12:01:32 CDT; 4min 19s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 3269 (sshd)
     Tasks: 1 (limit: 19003)
    Memory: 1.7M
       CPU: 12ms
   CGroup: /system.slice/ssh.service
           └─3269 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Jun 01 12:01:32 pop-os systemd[1]: Starting OpenBSD Secure Shell server...
Jun 01 12:01:32 pop-os sshd[3269]: Server listening on 0.0.0.0 port 22.
Jun 01 12:01:32 pop-os sshd[3269]: Server listening on :: port 22.
Jun 01 12:01:32 pop-os systemd[1]: Started OpenBSD Secure Shell server.
lines 1-16/16 (END)
```

Step 3: Find the IP Address

You can find the IP address in two ways:

- **From Proxmox Dashboard:** Under the VM summary tab
- **Inside Pop!_OS:**

```
ip a | grep inet
```

```
valid_lft forever preferred_lft forever
zippyb@pop-os:~$ ip a | grep inet
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
inet 192.168.1.151/24 brd 192.168.1.255 scope global dynamic noprefixroute e
np6s18
inet6 fd33:b9f1:a99c:144f:f8ee:22cb:15fe:edb5/64 scope global temporary dyna
mic
inet6 fd33:b9f1:a99c:144f:f3a2:e223:c714:dff0/64 scope global dynamic mngtmp
addr noprefixroute
inet6 fe80::6829:4e03:a9fb:d3b3/64 scope link noprefixroute
zippyb@pop-os:~$
```

Step 4: SSH From Another Machine

From your host machine or another computer on the LAN, connect:

```
ssh zippyb@192.168.1.151
```

You'll be prompted to accept the fingerprint and then enter your user password.

```
PS C:\Users\_\aonat> ssh zippyb@192.168.1.151
The authenticity of host '192.168.1.151 (192.168.1.151)' can't be established.
ED25519 key fingerprint is SHA256:7IzZhrE0VdUXcuOpUS+6kkoufXDhN6Mxtyp+g2IHtak.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.151' (ED25519) to the list of known hosts.
zippyb@192.168.1.151's password:
Welcome to Pop!_OS 22.04 LTS (GNU/Linux 6.12.10-76061203-generic x86_64)

* Homepage: https://pop.system76.com
* Support: https://support.system76.com

The programs included with the Pop!_OS system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Pop!_OS comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

zippyb@pop-os:~$
```

Optional: Use SCP to Transfer Files

```
scp file.txt zippyb@192.168.1.151:/home/nate/
```

Or use `rsync` for large/recurring syncs:

```
rsync -avz project/ zippyb@192.168.1.151:/home/nate/project/
```

Step 4: Enable and Configure UFW Firewall

1. Enable UFW:

```
sudo ufw enable
```

2. Allow SSH through the firewall:

```
sudo ufw allow ssh
```

3. Check firewall status:

```
sudo ufw status verbose
```

```
zippyb@pop-os:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
zippyb@pop-os:~$ sudo ufw allow ssh
Rule added
Rule added (v6)
zippyb@pop-os:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To Action From
--
22/tcp ALLOW IN Anywhere
22/tcp (v6) ALLOW IN Anywhere (v6)

zippyb@pop-os:~$
```

Done

I can now securely connect to my Pop!_OS VM using SSH for remote configuration and file transfers.

NVIDIA GPU Passthrough in Pop!_OS (Proxmox VE 8.2.7)

Date: June 1st, 2025

Category: Virtualization / GPU Passthrough

Backlink: [Setting Up SSH Access in POP_OS Proxmox](#)

This enable's GPU Passthrough as I'm going to use this VM for Local AI projects

Step 1: Enable IOMMU in Proxmox

Edit the GRUB configuration:

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

Set this line for AMD CPUs:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet amd_iommu=on"
```

Update GRUB and reboot:

```
update-grub  
reboot
```

Step 2: Bind GPU to `vfio-pci`

Get the NVIDIA GPU and related device IDs:

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

Example output:

```
07:00.0 VGA compatible controller [10de:1e84]
07:00.1 Audio device [10de:10f8]
07:00.2 USB controller [10de:1ad8]
07:00.3 Serial bus controller [10de:1ad9]
```

Bind them to `vfio-pci`:

```
echo "options vfio-pci ids=10de:1e84,10de:10f8,10de:1ad8,10de:1ad9" > /etc/modprobe.d/vfio.conf
```

Then:

```
update-initramfs -u
reboot
```

Step 3: Confirm vfio-pci Binding

Check if vfio-pci is now in use:

```
lspci -nnk | grep -A 3 -i nvidia
```

You should see something like:

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

Step 4: Attach GPU to Pop!_OS VM

In the **Proxmox Web GUI**:

1. Power off the Pop!_OS VM.
2. Go to `Hardware > Add > PCI Device`.
3. Add only the **VGA compatible controller** (e.g., `07:00.0`).
4. Enable the following checkboxes:
 - ROM-Bar
 - Primary GPU

- Optional: You can try passing the other 3 GPU functions (Audio, USB, Serial Bus) if needed.

Start the VM.

Step 5: Verify GPU Access in Pop!_OS

Inside the VM, run:

```
nvidia-smi
```

To check for CUDA:

```
nvcc --version
```

(Optional if GUI is installed):

```
glxinfo | grep "OpenGL renderer"
```

Completion Notes

- GPU passthrough is functional.
- Host Proxmox system is stable and still accessible via Web GUI.
- Pop!_OS VM is now accelerated using the RTX 2070 SUPER.
- SSH and firewall (UFW) are set up on the guest.