We need to configure the Proxmox host to pass the PCI device through to VMs:

- **Edit GRUB for IOMMU:** SSH into the Proxmox host or use the shell. Edit `/etc/default/grub`. Find the `GRUB_CMDLINE_LINUX_DEFAULT` line and append the following parameters (depending on CPU vendor):
- For Intel: `intel_iommu=on iommu=pt`
- For AMD: `amd_iommu=on iommu=pt`
  This ensures IOMMU is active and in passthrough mode. Save the file and then run `update-grub` to apply changes.
- **Load VFIO Modules:** Edit (or create) `/etc/modules` and add these lines:

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

  This makes sure the VFIO (Virtual Function I/O) driver is loaded on boot, which is needed to attach devices to VMs.
- **Blacklist Host Drivers:** Since the host doesn't need to use the GPU directly, we prevent the host OS from grabbing it. For NVIDIA, blacklist the nouveau driver (open-source) and possibly the NVIDIA driver if it's installed. To do so, create a file like `/etc/modprobe.d/blacklist-nouveau.conf` with content `blacklist nouveau` (and similarly blacklist `nvidia`, `nvidia_drm`, etc. if any). Additionally, it's recommended to prevent the EFI framebuffer from locking the GPU, by adding `video=efifb:off` to the GRUB cmdline if needed.
- **Bind GPU to VFIO:** Optionally, you can specify that the GPU's IDs should be bound to vfio-pci driver early. One method: create a config at `/etc/modprobe.d/vfio.conf` with:

```
options vfio-pci ids=<GPU_ID>,<Audio_ID>
```

  Replace `<GPU_ID>` and `<Audio_ID>` with the vendor:device IDs of your GPU's functions (you can get these from `lspci -n` output, e.g., `10de:1eb8,10de:10f0` format for NVIDIA GPU and its audio). This ensures that on boot, those PCI devices attach to vfio-pci driver instead of any host driver.

After these changes, **reboot the Proxmox host**. When it comes back, verify that the GPU is now using the vfio driver: `lspci -k -s 01:00.0` should show it bound to `vfio-pci` (and not nouveau or nvidia). Also, run `dmesg | grep -e IOMMU -e DMAR` to confirm IOMMU is enabled with no errors.

3. VM Configuration in Proxmox:

- If not already done, create a new VM (for example, a Windows 11 VM or an Ubuntu VM depending on need) via the Proxmox web interface. Give it the proper UEFI BIOS (OVMF) if using UEFI, and add the **Qemu Guest Agent** for convenience (especially for Windows).
- **Add the PCI Device:** In the Proxmox web UI, go to the hardware configuration of the VM. Click **Add -> PCI Device**. From the dropdown, select your GPU (it will show up as something like `01:00.0 NVIDIA Corporation ...`). Also, **add the GPU's audio function** (usually `01:00.1`) as a second PCI Device in the VM. When adding, check the boxes for **"PCI-Express"** and **"All Functions"** if available (Proxmox 7/8 allows a checkbox for "All Functions" which ensures multi-function devices like GPUs with audio are handled together).
- **Machine Type and CPU:** It's often recommended to set the VM's BIOS to OVMF (UEFI) and Machine type to `q35` (which supports PCIe passthrough better). For CPU type, use **host** (to expose all CPU features to the VM). And crucially, for an NVIDIA card being passed to a Windows VM, add the following **args** in the VM options (in Proxmox GUI, under VM -> Options -> "args" or via editing `/etc/pve/qemu-server/<vmid>.conf`):

  ```
  args: -cpu host,hidden=1
  ```

  The `hidden=1` part hides the hypervisor signature from the GPU driver; this avoids the infamous "Error 43" that NVIDIA consumer GPUs show in VMs by design. Our guide explains this and ensures it's set.
- **Memory & HugePages:** If the GPU has a lot of VRAM or if doing heavy compute, we advise enabling hugepages for the VM memory. In the VM's Memory settings, you might set "hugepages = 2048" (2MB huge pages) and ensure the host has enough free hugepages. This can improve performance and reduce any memory mapping issues.

4. Install Drivers in VM:

Start up the VM. If it's Windows, on first boot after adding the GPU, it might detect new hardware. You'll need to install the official NVIDIA drivers inside the VM (download from NVIDIA's site). For Linux VMs, install the appropriate NVIDIA proprietary driver package. After installation, reboot the VM. Then verify within the VM that the GPU is working: - On Windows, check Device Manager: the GPU should appear under Display Adapters with no error icon. Run something like GPU-Z or the NVIDIA Control Panel to see that it's functioning. If using it for display, you can attach a virtual monitor or use Parsec/RDP to see GPU output, but if headless for compute, just ensure it's recognized. - On Linux, run `nvidia-smi` in the VM to see if it communicates with the card and shows stats.

5. Testing and Troubleshooting:

We recommend a test workload: e.g., run a quick benchmark or, for AI, try running a model using the GPU (from within the VM). Monitor Proxmox tasks to ensure no errors like IOMMU faults. Common issues and solutions are noted in the guide: - If the VM fails to start and you see an IOMMU error, the device might be in a group that isn't isolated. Sometimes, adding `pch_reset=1` or similar args for certain motherboards is needed, or making sure no other device in the IOMMU group is in use. Our guide suggests using `find /sys/kernel/iommu_groups/ -type l` to see groupings. - Error 43 in Windows device manager means

the Proxmox host uses UEFI, sometimes the GPU is bound to efifb. We already suggested `video=efifb:off` to grub; if not, do that and reboot. - If you need to free the GPU for host use occasionally, you would unload vfio and load nvidia drivers; but in general, we dedicate the GPU to the VM full-time in our deployments.

6. Advanced:

The guide also briefly covers advanced scenarios: - **Multiple VMs with one GPU:** Not generally possible simultaneously unless using NVIDIA vGPU (not covered here due to licensing) or Intel GVT-g/AMD SR-IOV if supported. Our approach is one GPU per critical VM. If multiple lesser VMs need acceleration, consider time-slicing or an alternate approach (this is not typical in our client setups; usually one powerful VM uses the GPU for AI or a virtual desktop).
- **GPU for Linux acceleration:** If the VM is Linux and headless (for compute), no special steps beyond driver install are needed. If it's for a Linux desktop VM (rare in our case), we ensure Spice/virtio graphics are off and maybe use a dummy HDMI plug on the GPU to trick it if needed for proper resolution.

By following the **Proxmox GPU Passthrough Guide**, our internal team (or the client's IT) can successfully assign a physical GPU to a VM. This unleashes powerful capabilities: for example, a Windows VM can be used for video editing or CAD with GPU, or our Ubuntu server VM can run machine learning tasks at dramatically improved speeds. The process is intricate but our guide breaks it down to manageable steps, ensuring a smooth setup in line with our overall system architecture.

TrueNAS Scale Configuration

This guide provides a complete setup walkthrough for **TrueNAS SCALE**, which is our choice for on-site network-attached storage (NAS) and lightweight container hosting. We detail how to configure ZFS storage pools for optimal performance and reliability, how to set up network shares, and how to deploy applications on TrueNAS SCALE. Following this guide ensures the storage system is rock-solid and tuned for our clients' needs.

1. Installation & Initial Access:

Assuming TrueNAS SCALE is freshly installed on the NAS server (either bare-metal or as a VM with direct disk access), connect to the web UI via a browser (`https://<truenas-ip>/`). The first time setup wizard will prompt for basics: - **Network Config:** Assign a static IP address in the management VLAN (as per the Blueprint). Make sure DNS settings point to the local DNS server (or your router) and that the gateway is set correctly. Enable the web interface over HTTPS for security (TrueNAS can generate a self-signed cert or you can install a proper certificate).
- **Admin Account:** Set a strong root password (our practice: use a generated passphrase, store it in the password manager). Optionally, create a non-root admin user for daily use and disable root login over SSH for security. Ensure SSH is enabled only if needed (and if so, key-based auth is a must).

2. Creating ZFS Pool(s):