Installation

Installation Host

Installation von Proxmox VE 9

1. ISO herunterladen

- Offizielle Downloadseite:

  https://www.proxmox.com/en/downloads
- Gewählt wird: **Proxmox VE 9.x ISO Installer** (aktuell `proxmox-ve_9.x.iso`).

---

2. Bootfähigen USB-Stick unter Windows erstellen

Am einfachsten mit **Rufus**:

1. Download Rufus: https://rufus.ie
2. USB-Stick (≥ 4 GB, wird gelöscht) einstecken.
3. Rufus starten → folgendes einstellen:
   - **Gerät**: USB-Stick auswählen
   - **Boot-Auswahl**: heruntergeladene `proxmox-ve_9.x.iso`
   - **Partitionstyp**: GPT
   - **Zielsystem**: UEFI (nicht BIOS/Legacy)
   - Rest Standard lassen
4. Start → ISO wird auf den Stick geschrieben.

---

3. Installation von Proxmox VE

1. Server einschalten und vom USB-Stick booten (UEFI-Modus wählen).
2. Installationsmenü → `Install Proxmox VE` wählen.
3. Lizenzbedingungen akzeptieren.
4. **Zieldatenträger auswählen**:
   - Samsung NVMe 990 Pro 2 TB (Systemplatte).

    - Dateisystem: **ext4** (einfach und stabil, ZFS nur bei RAID oder Snapshots nötig).
5. Zeitzone: `Europe/Berlin`, Tastaturlayout `de`.
6. Root-Passwort und E-Mail-Adresse setzen (für Benachrichtigungen).
7. Netzwerkeinstellungen:
    - Hostname: `pve.localdomain` (später anpassen).
    - Management-IP manuell vergeben.
8. Installation starten, nach Abschluss neustarten.

---

4. Erste Anmeldung

- Webinterface über: `https://<IP-des-Servers>:8006`
- Login mit `root` und dem gesetzten Passwort.

---

5. BIOS-Anpassungen für GPU-Passthrough

Vor dem Start der VMs müssen im BIOS folgende Optionen aktiviert werden:

- **Intel VT-d** → `Enabled`
- **SR-IOV** → `Enabled`
- **Above 4G Decoding** → `Enabled`
- **Resizable BAR** → `Enabled` (falls vorhanden, für GPU Performance)
- **Primary Display** → auf `iGPU` oder `Onboard` stellen (damit die NVIDIA-Karte für Passthrough frei ist).

---

6. GRUB & Kernel-Anpassungen für Passthrough

Auf dem Proxmox-Host anmelden (SSH oder Shell).

a) IOMMU aktivieren

Datei `/etc/default/grub` bearbeiten:

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

Die Zeile mit `GRUB_CMDLINE_LINUX_DEFAULT` anpassen:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt"
```

Speichern und GRUB neu generieren:

```
update-grub
```

b) VFIO-Module laden

Datei `/etc/modules` bearbeiten und folgende Zeilen ergänzen:

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

---

7. GPU identifizieren & binden

1. PCI-Geräte auflisten:

   ```
   lspci -nn
   ```

   → GPU (z. B. `10de:2805`) und Audio-Controller (`10de:228b`) notieren.
2. Datei `/etc/modprobe.d/vfio.conf` erstellen:

   ```
   options vfio-pci ids=10de:2805,10de:228b
   ```
3. NVIDIA-Treiber auf Host blockieren:
   Datei `/etc/modprobe.d/blacklist.conf` ergänzen:

   ```
   blacklist nouveau
   blacklist nvidia
   blacklist nvidiafb
   ```
4. Initramfs neu erstellen:

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

---

8. Neustart & Kontrolle

Nach Neustart prüfen, ob die GPU an VFIO gebunden ist:

```
lspci -nnk | grep -A 3 -E "10de"
```

→ Sollte `Kernel driver in use: vfio-pci` anzeigen.

Einrichten VM

Installation Ubuntu 24.04.3 LTS (Minimal Server)

1. ISO herunterladen

Offizielle Quelle:

https://ubuntu.com/download/server

- Variante: **Ubuntu Server 24.04.3 LTS**
- Image: `ubuntu-24.04.3-live-server-amd64.iso`

---

2. VM in Proxmox anlegen

1. In der Proxmox Web-GUI: **Create VM**
   - Name: `KI-VM`
   - ISO: `ubuntu-24.04.3-live-server-amd64.iso` (zuvor hochgeladen in Proxmox Storage)
   - System:
     - BIOS: **OVMF (UEFI)**
     - Machine: **q35**
     - Graphic Card: **none** (GPU wird durchgereicht)
   - Disks:
     - Speicher auf **Crucial NVMe 4 TB** oder System-SSD je nach Planung
     - Größe: mind. 250 GB (je nach Modellgrößen)
     - Cache: **Write back (unsafe)** (für Performance)
   - CPU:
     - 6 Cores (i5-13400 hat 10 → 6 reichen für KI-Workloads, Rest für Host)
     - Typ: **host**
   - RAM:
     - 32 GB (von 128 GB gesamt)
     - Ballooning: deaktivieren (konstante Zuweisung)
   - Netzwerk: **VirtIO (paravirtualized)**
   - Hardware: **GPU & Audio-Device** via PCI Passthrough hinzufügen (wie vorher eingerichtet).

3. Ubuntu Installation

1. Boot von ISO → Auswahl: **Install Ubuntu Server**
2. Sprache: `Deutsch` (oder Englisch, je nach Vorliebe)
3. Tastatur: `Deutsch`
4. Netzwerk:
   - `ens18` (VirtIO) automatisch via DHCP oder manuell konfigurieren
   - Empfehlung: **statische IP** für die VM (z. B. 192.168.33.200)
5. Storage:
   - Geführte Installation auf die virtuelle Disk
   - Partitionierung: Standard (LVM möglich, aber nicht zwingend)
6. Benutzer anlegen:
   - Username: `kiadmin`
   - Passwort: sicher setzen
   - SSH-Server installieren: **Ja**
7. Snap-Pakete: alle abwählen (nicht benötigt)
8. Installation starten → Neustart nach Abschluss

---

4. Erste Anpassungen nach der Installation

Nach Login per SSH oder Console:

a) System aktualisieren

```
sudo apt update && sudo apt upgrade -y
sudo reboot
```

b) Nützliche Basis-Tools installieren

```
sudo apt install -y htop ncdu git curl wget unzip zip tar net-tools iftop lsb-release pciutils
```

c) SSH konfigurieren

```
sudo nano /etc/ssh/sshd_config
```

Empfohlen:

- `PermitRootLogin no`
- `PasswordAuthentication no` (falls SSH-Key genutzt wird)

Dann:

```
sudo systemctl restart ssh
```

d) Zeitsynchronisation prüfen

```
timedatectl set-timezone Europe/Berlin
timedatectl status
```

---

5. GPU-Treiber vorbereiten

Da die GPU durchgereicht wird, braucht die VM die NVIDIA-Treiber:

a) Repository aktivieren

```
sudo apt install -y software-properties-common
sudo add-apt-repository ppa:graphics-drivers/ppa -y
sudo apt update
```

b) NVIDIA-Treiber installieren

```
sudo apt install -y nvidia-driver-550 nvidia-utils-550
```

c) Neustart & Test

```
nvidia-smi
```

→ sollte GPU-Daten (Modell RTX 5060 Ti, Treiber, CUDA-Version) anzeigen.

---

6. Optional: Basis-Optimierungen für KI-Workloads

- **Swappiness reduzieren** (damit RAM bevorzugt genutzt wird):

  ```
  echo 'vm.swappiness=10' | sudo tee -a /etc/sysctl.conf
  sudo sysctl -p
  ```

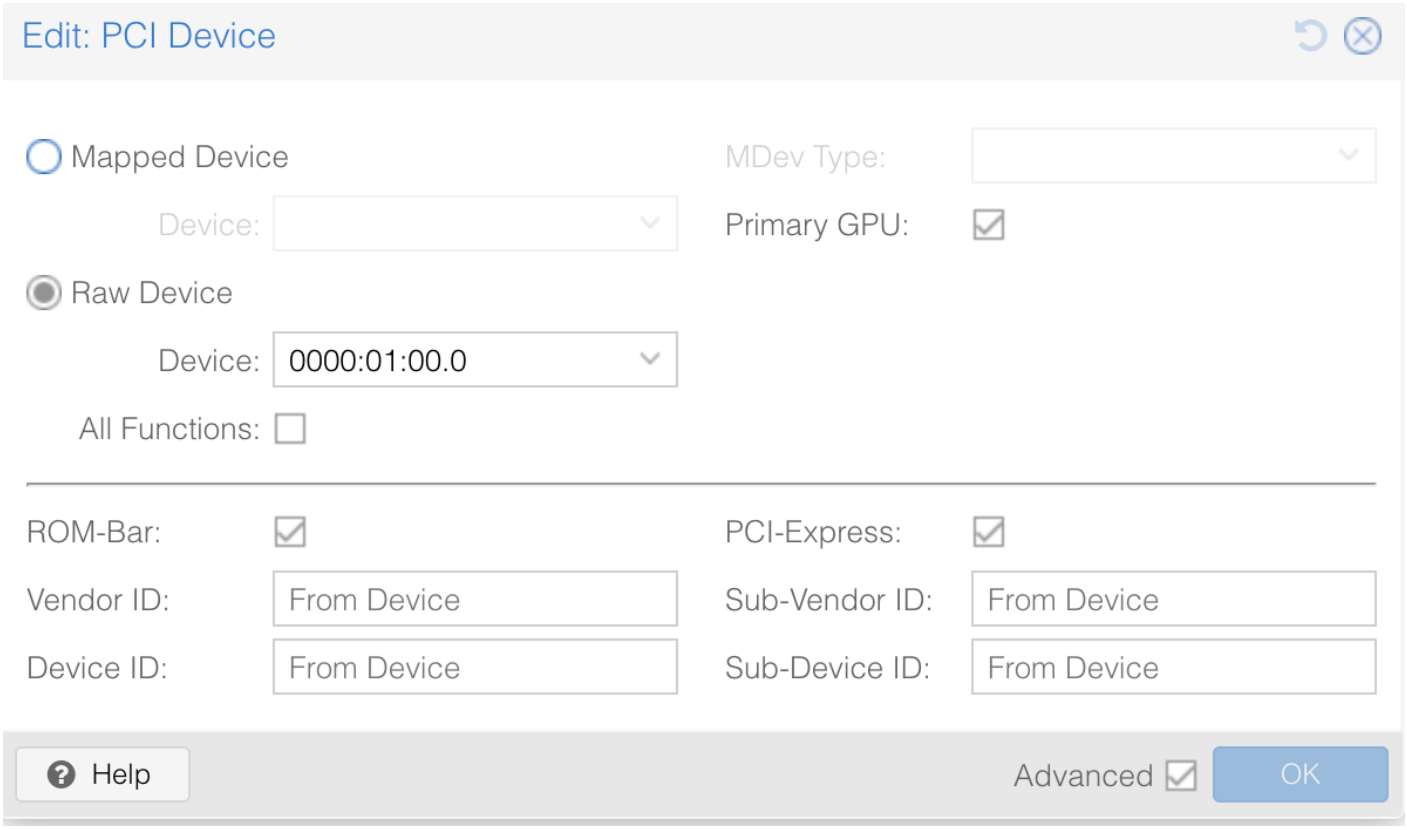
- **Transparent Hugepages deaktivieren** (manchmal Performance-Gewinn bei LLMs):

  ```
  echo never | sudo tee /sys/kernel/mm/transparent_hugepage/enabled
  ```

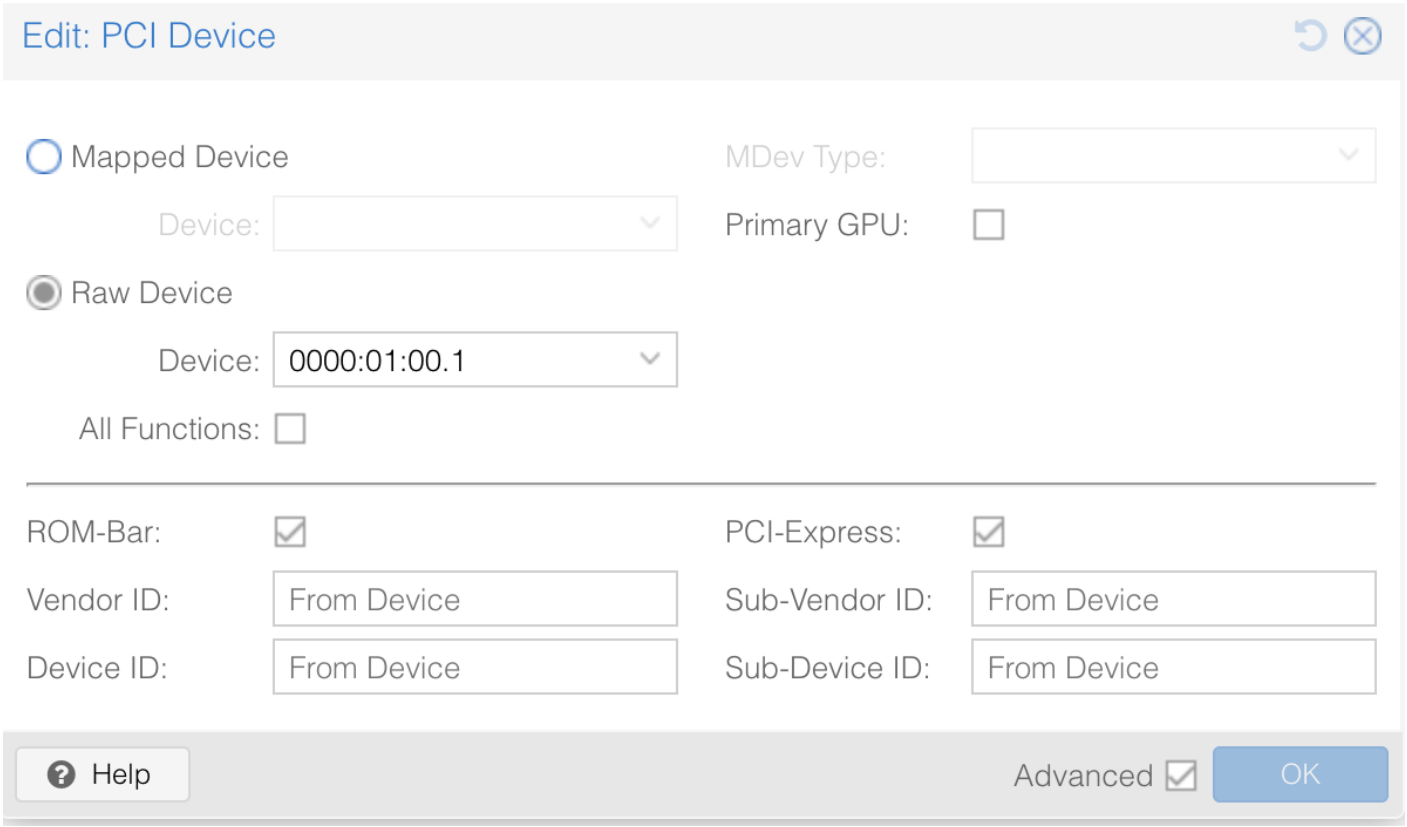
- **ufw aktivieren** (Firewall, falls nicht durch Docker geregelt):

  ```
  sudo apt install ufw -y
  sudo ufw default deny incoming
  sudo ufw default allow outgoing
  sudo ufw allow ssh
  sudo ufw enable
  ```

Konfiguration GPU:



Konfiguration GPU-Audio:


Edit: PCI Device
Mapped Device
MDev Type:
Device:
Primary GPU:
Raw Device
Device: 0000:01:00.1
All Functions:
ROM-Bar:
PCI-Express:
Vendor ID:
From Device
Sub-Vendor ID:
From Device
Device ID:
From Device
Sub-Device ID:
From Device
Help
Advanced
OK

Installation Docker

Installation von Docker & Docker Compose

1. Alte Versionen entfernen (falls vorhanden)

Vorherige Docker-Installationen oder Reste löschen:

```
sudo apt remove -y docker docker-engine docker.io containerd runc
```

---

2. Abhängigkeiten installieren

```
sudo apt update
sudo apt install -y ca-certificates curl gnupg lsb-release
```

---

3. Docker GPG-Key hinzufügen

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

---

4. Docker-Repository hinzufügen

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
```

```
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

---

5. Docker Engine installieren

```
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

---

6. Installation prüfen

```
docker --version
docker compose version
```

---

7. Benutzerrechte anpassen

Standardmäßig braucht man `sudo` für Docker. Damit normale Nutzer (z. B. `kiadmin`) Docker nutzen können:

```
sudo usermod -aG docker kiadmin
```

→ Danach **ab- und wieder anmelden**.

---

8. Docker als Dienst aktivieren

Damit Docker beim Booten automatisch startet:

```
sudo systemctl enable docker
sudo systemctl start docker
```

---

9. Test mit Hello-World Container

```
docker run hello-world
```

→ Sollte eine Bestätigungsmeldung ausgeben.

---

10. (Optional) Standard-Speicherpfad anpassen

Falls Container und Images nicht auf der Systemplatte, sondern auf einer dedizierten NVMe liegen sollen:

1. Docker-Dienst stoppen:

   ```
   sudo systemctl stop docker
   ```
2. Konfigurationsdatei erstellen:

   ```
   sudo mkdir -p /etc/docker
   echo '{
     "data-root": "/opt/docker"
   }' | sudo tee /etc/docker/daemon.json
   ```
3. Verzeichnis anlegen & Rechte setzen:

   ```
   sudo mkdir -p /opt/docker
   sudo chown -R root:docker /opt/docker
   ```
4. Docker neu starten:

   ```
   sudo systemctl start docker
   ```