Proxmox

- Proxmox installieren
- SMB Share einrichten
- OVA Datei in Proxmox einbinden
- VM Festplatte vergrößern
- Windows VM in Proxmox
- Proxmox Terminal
- USB an VM verbinden
- VM mit Cronjob starten
- Installieren von Raspiblitz auf Proxmox
- Auf Proxmox USB durchschleifen für FHEM
- Raspiblitz auf Proxmox installieren
- Festplattenübersicht
- LXC Container Fix Docker Problem
- Debian Template erstellen
- Backup Proxmox

Proxmox installieren

Proxmox Update:

Unter Proxmox Updates: Aktualisieren und Upgrade ausführen

Befehle die zusätzlich eingegeben werden müssen

```
# Source-Liste anpassen
nano /etc/apt/sources.list
# der inhalt ist gegen den aus der liste auzutauschen:
https://pve.proxmox.com/wiki/Package_Repositories#sysadmin_no_subscription_repo

# PVE Enterprise.list ändern
nano /etc/apt/sources.list.d/pve-enterprise.list
erste zeile auskommentieren

# Updates Installieren
apt update && apt dist-upgrade -y

Subscription Meldung deaktivieren
https://dannyda.com/2020/05/17/how-to-remove-you-do-not-have-a-valid-subscription-for-this-server-from-proxmox-virtual-environment-6-1-2-proxmox-ve-6-1-2-pve-6-1-2/
```

Zweite Festplatte einbinden

0. Herunterfahren und Festplatte einbauen
1. unter Proxmox Disks werden die Festplatten angezeigt.
2. Festplatte einrichten
    0. apt install parted
    1. lsblk # alle festplatten anzeigen
    2. parted /dev/sda "mklable gpt" # Festplatte wird formatiert aufpassen das die richtige gewählt ist
3. proxmox > Verzeichnis: Erstellen Directory > Disk auswählen > Dateisystem ext4 > Name > Erstellen

Windows installieren

https://www.youtube.com/embed/-M-F2LKC7L0

- Festplatte: vioscsi
- Netzwerkkarte: netkvm

SSH Verbindung herstellen

Muss in der config datei zugelassen werden:

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

SMB Share einrichten

Richte als Erstes auf deinem NAS eine SMB Freigabe für den CT Template und ISO Share ein. Außerdem ist es zu empfehlen, einen eigenen User für den Share anzulegen, der nur auf deinem Proxmox Node zum Einsatz kommt. SMB Share auf Proxmox mit fstab mounten

Wenn du den Share erstellt hast, kannst du nun auf deinem Proxmox Server (Node) die SMB Freigabe mithilfe von fstab mounten, sodass diese auch immer beim Neustart des Proxmox Nodes aktiv wird. Credentials File erstellen

Erstelle als erstes das Credentials File. Öffne die Datei mit dem folgenden Befehl.

`nano /home/.smbcreds`

Füge folgenden Codeblock in die Date ein. Ersetze „deinbenutzername“ mit dem auf deinem NAS vergebenen Benutzernamen und „deinpasswort“ mit dem Passwort des NAS Benutzers.

```
username=deinbenutzername
password=deinpasswort
```

Share mit fstab mounten

Rufe fstab mithilfe des folgenden Befehls auf.

`nano /etc/fstab`

Füge am Ende der Datei folgenden Codeblock ein. Ersetze dabei

„192.168.100.100“ mit der IP Adresse deines NAS
„ctshare“ mit dem Sharenamen auf deinem NAS

```
//192.168.100.100/ctshare /mnt/data/ctshare cifs credentials=/home/.smbcreds,uid=1000,gid=1000 0 0
```

Mountpoint erstellen

Erstelle nun als letzen Schritt im Mount-Vorgang den Mountpoint. Dieser ist der lokale Ordner auf deinem Proxmox Node, in den der oben angegebene NAS-Share gemountet wird.

```
mkdir /mnt/data && mkdir /mnt/data/ctshare
```

OVA Datei in Proxmox einbinden

Virtualisierte Betriebssysteme können zum Beispiel aus VirtualBox im OVA-Format (Open Virtual Appliance) exportiert werden. Sie beinhalten neben der virtuellen Festplatte im VMDK-Format (Virtual Machine Disk) eine Beschreibung der Einstellungen im OVF-Format (Open Virtualization Format). Darin sind zum Beispiel Informationen über die Hardwareeinstellungen, wie die Anzahl der verwendeten Kerne und die Größe des Arbeitsspeichers enthalten.

Diese Anleitung beschreibt, wie ein virtualisiertes System als OVA-Datei zu Proxmox migriert werden kann:

1. Virtuelle Maschine anlegen

Als Erstes muss eine neue virtuelle Maschine angelegt werden. Hier können wie gewohnt die Hardware-Daten konfiguriert werden. Wichtig ist nur, dass keine Festplatte angelegt wird. Die ID der neuen Maschinen wird bei dem späteren Import dann noch benötigt.

2. Datei auf den Proxmox Server kopieren

Als Nächstes wird die OVA-Datei auf den Server kopiert. Dies kann zum Beispiel mit Filezilla erfolgen. Als Verbindungsdaten muss die IP-Adresse, der Benutzernamen root und das dazugehörige Passwort eingegeben werden. Die Ablage kann im root-Verzeichnis erfolgen, da die Dateien nach einer erfolgreichen Migration gelöscht werden können.

3. OVA-Datei entpacken

Bei der OVA-Datei handelt es sich eigentlich um ein tar-Archiv. Daher muss es erst einmal entpackt werden:

```
tar -xvf windows.ova
```

Als Ergebnis stehen nun zwei Dateien zur Verfügung. In diesem Beispiel heißen sie windows.vmdk und windows.ovf.

4. Konvertieren und Importieren des Laufwerks

Nun muss die VMDK-Datei konvertiert und der virtuellen Maschine zugeordnet werden. Dazu verwenden wir das Tool bzw. den Befehl qm von Proxmox. Nach der Anweisung importdisk kommt die ID der neuen Maschine und dahinter der Dateinamen der VMDK-Datei. Anschließend erfolgt der Speicherort, dieser ist links im Menü von Proxmox sichtbar, er lautet häufig local. Zum Abschluss wird noch die Angabe des Zielformates mit -format qcow2 angegeben, qcow2 ist mittlerweile das Standardformat von Proxmox.

```
qm importdisk 100 windows.vmdk local -format qcow2
```

5. Einbinden des Laufwerks

Bei der Maschine sollte nun unter Hardware ein neuer Eintrag hinzugefügt worden sein. In meinem Fall das Laufwerk „Unused Disk 0“. Mit einem Doppelklick auf den Eintrag erscheint ein Fenster, um das Laufwerk hinzuzufügen. Danach kann die virtuelle Maschine in Proxmox gestartet werden.

6. Starten der Maschine

Jetzt ist in proxmox soweit alles eingerichtet und die virtuelle Maschine kann gestartet werden.

Häufige Fehlerquelle

Hin und wieder kann es zu einem Fehler kommen, dass die Meldung „No bootable device found“ erscheint. Dies kann an einer Inkompatibilität der Festplatten-Schnittstelle liegen. Um dies zu beheben, muss die Festplatte mit „Detach“ entfernt werden und beim erneuten Einbinden bei der Option BUS/Device der Eintrag SCSI ausgewählt werden. Danach muss unter Options noch die „Boot Order“ Einstellung überprüft werden. Hier kann noch der alte Eintrag der Schnittstelle stehen und so wieder einen Start verhindern. Funktioniert SCSI nicht, dann einfach mit SATA oder IDE probieren. Damit hat bei mir die Migration geklappt.

VM Festplatte vergrößern

Festplatte vergrößern Proxmox

Unter Hardware
Disk Action > Resize disk

> Kann immer wieder vergrößert werden. Verkleinern geht nicht

Festplatte anzeigen

```
lsblk
```

Befehle zum Vergrößern

Heraussuchen welche Partition verändert werden soll

```
growpart /dev/sda 2
```

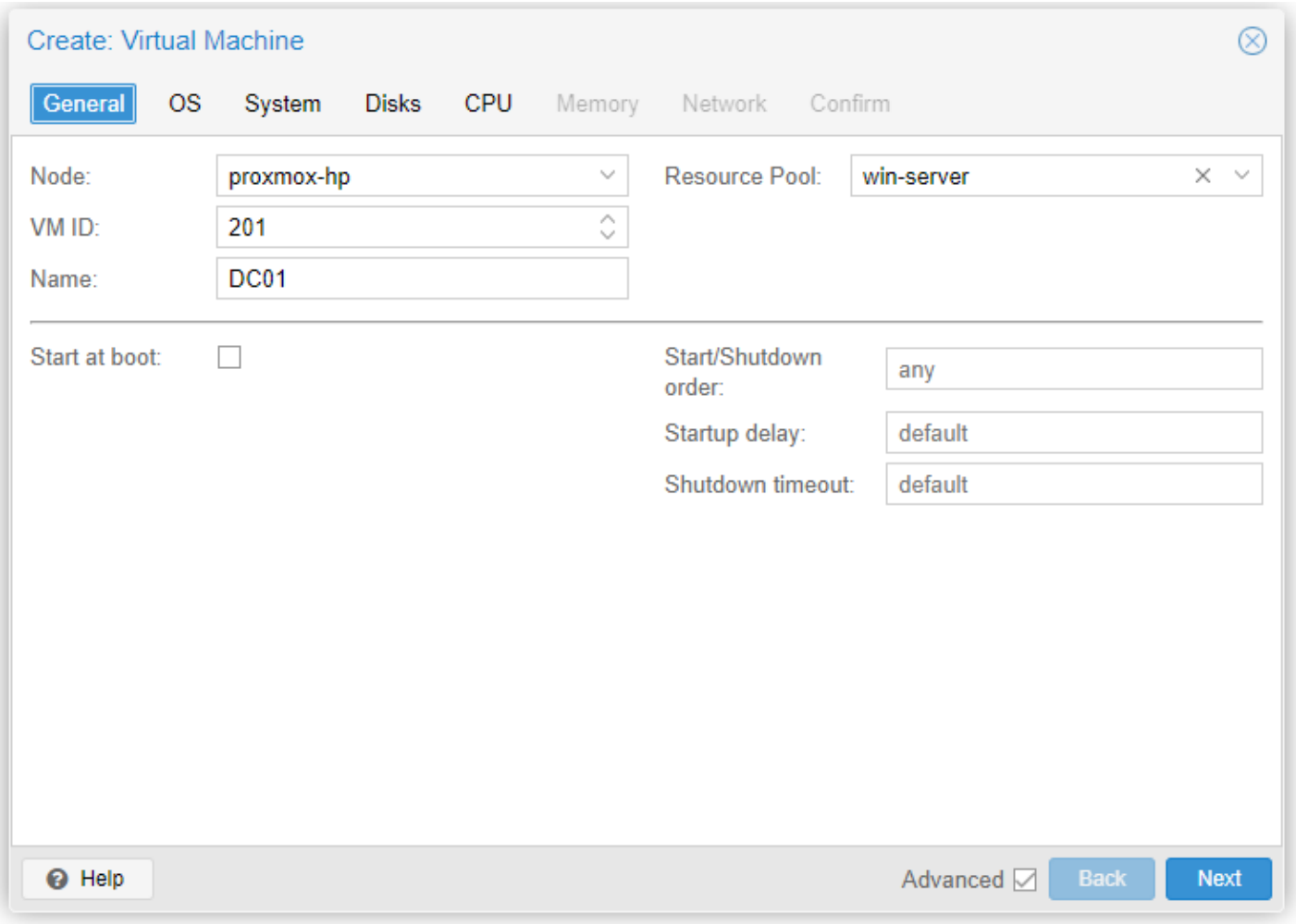
die Komplette Größe der Partition zuweisen

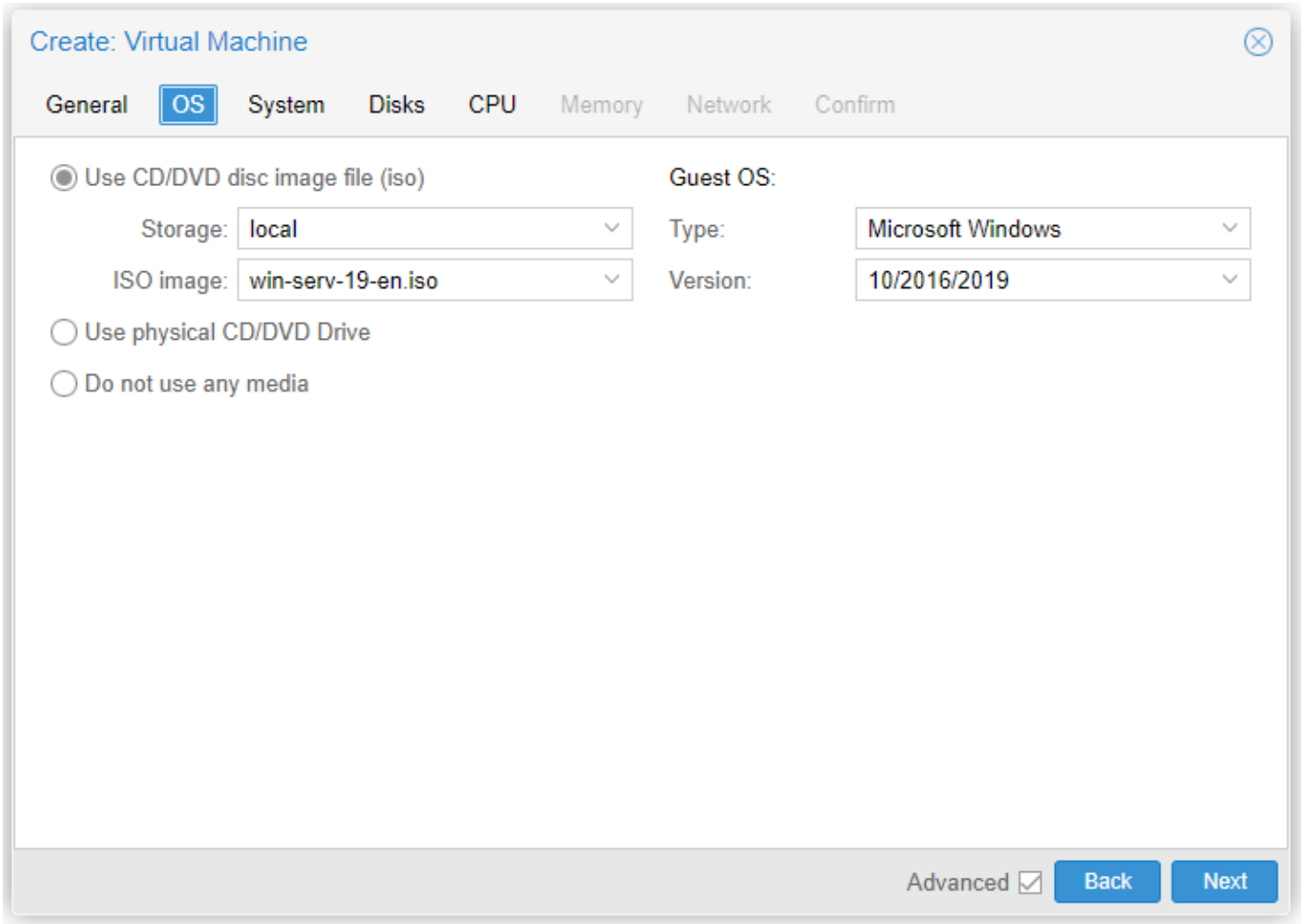
```
sudo lvextend -l 100%VG ubuntu-vg/ubuntu-lv
```

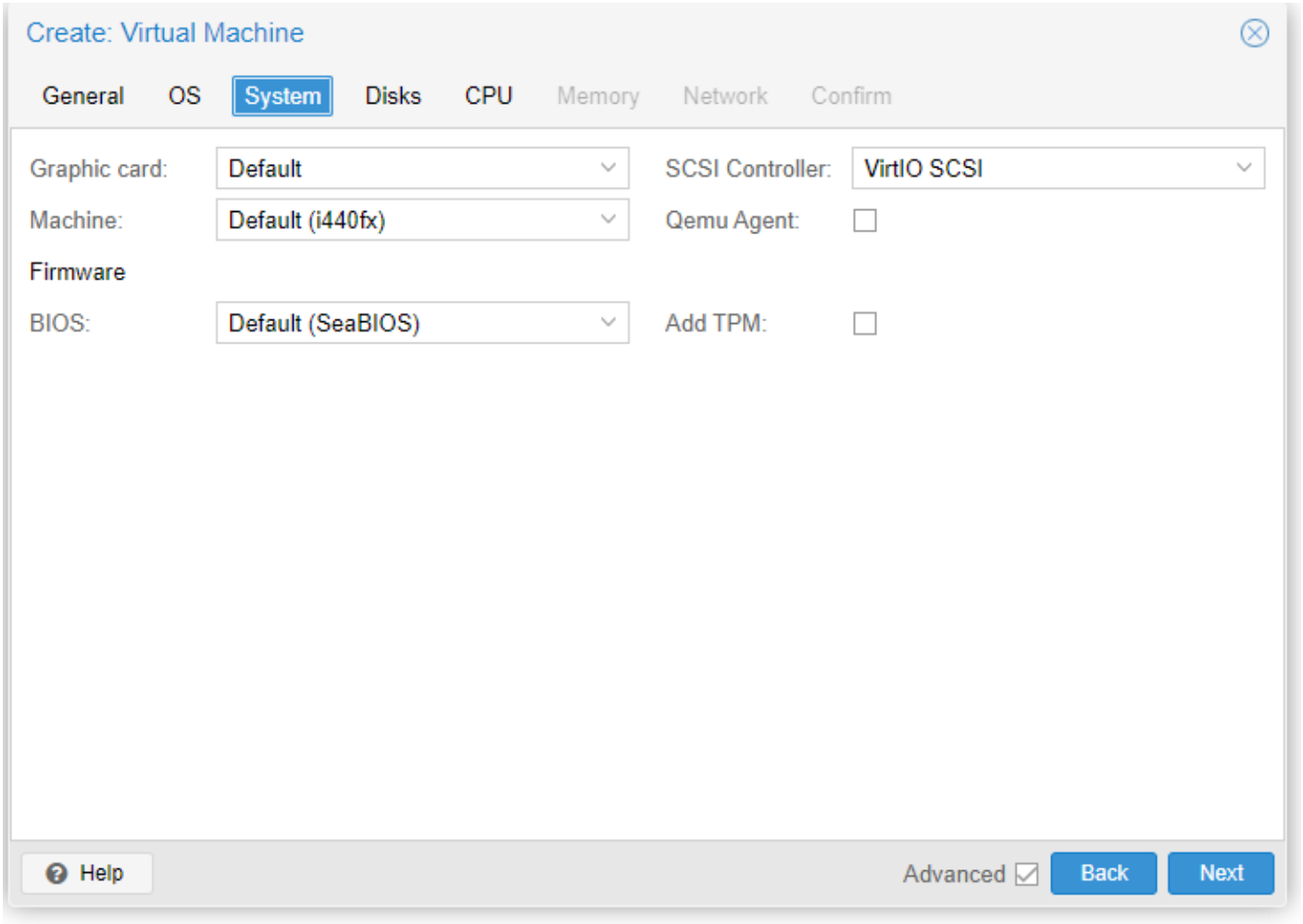
```
resize2fs /dev/ubuntu-vg/ubuntu-lv
```

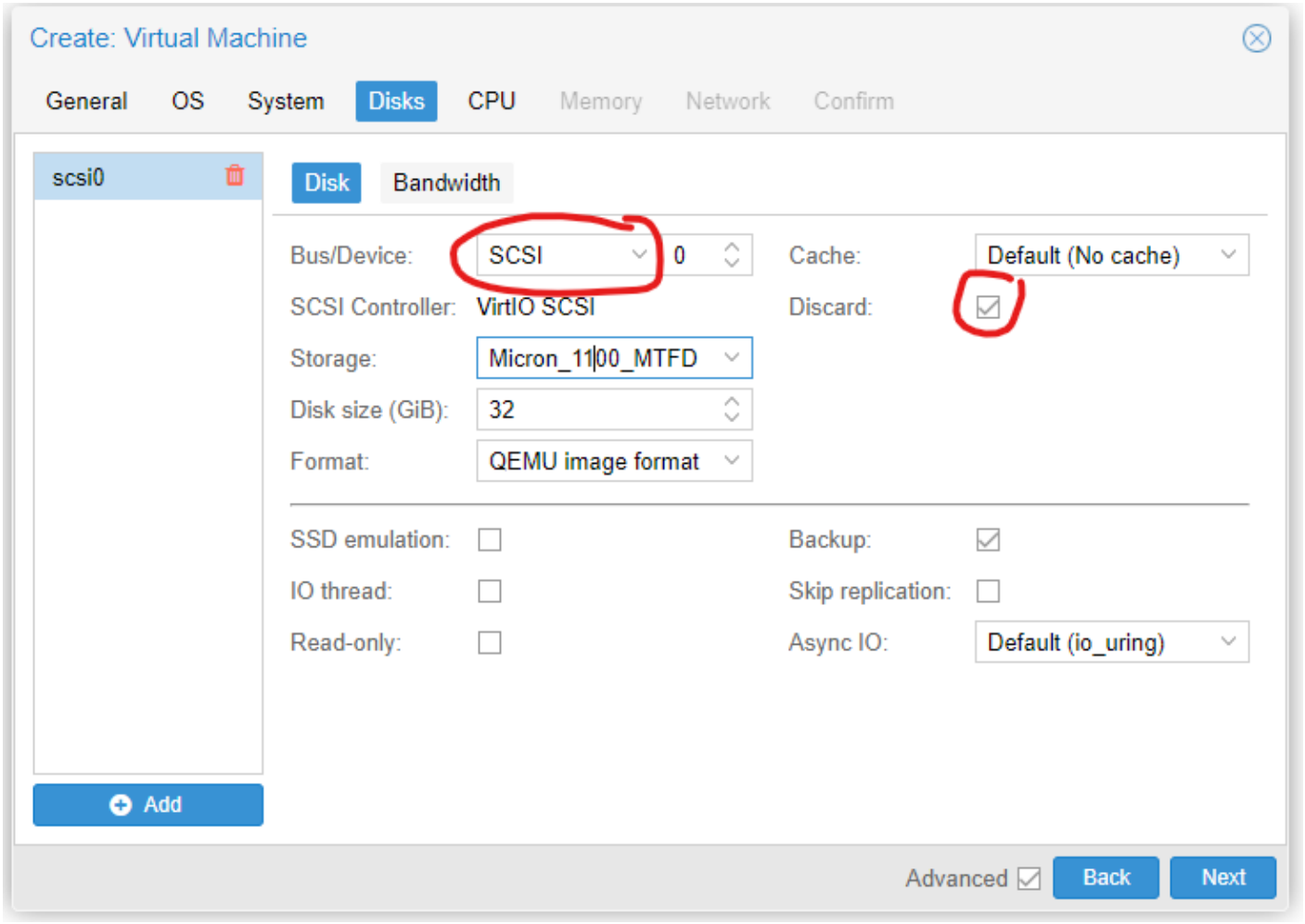
Windows VM in Proxmox

Wie erstelle ich eine VM in Proxmox mit Windows Server die läuft?

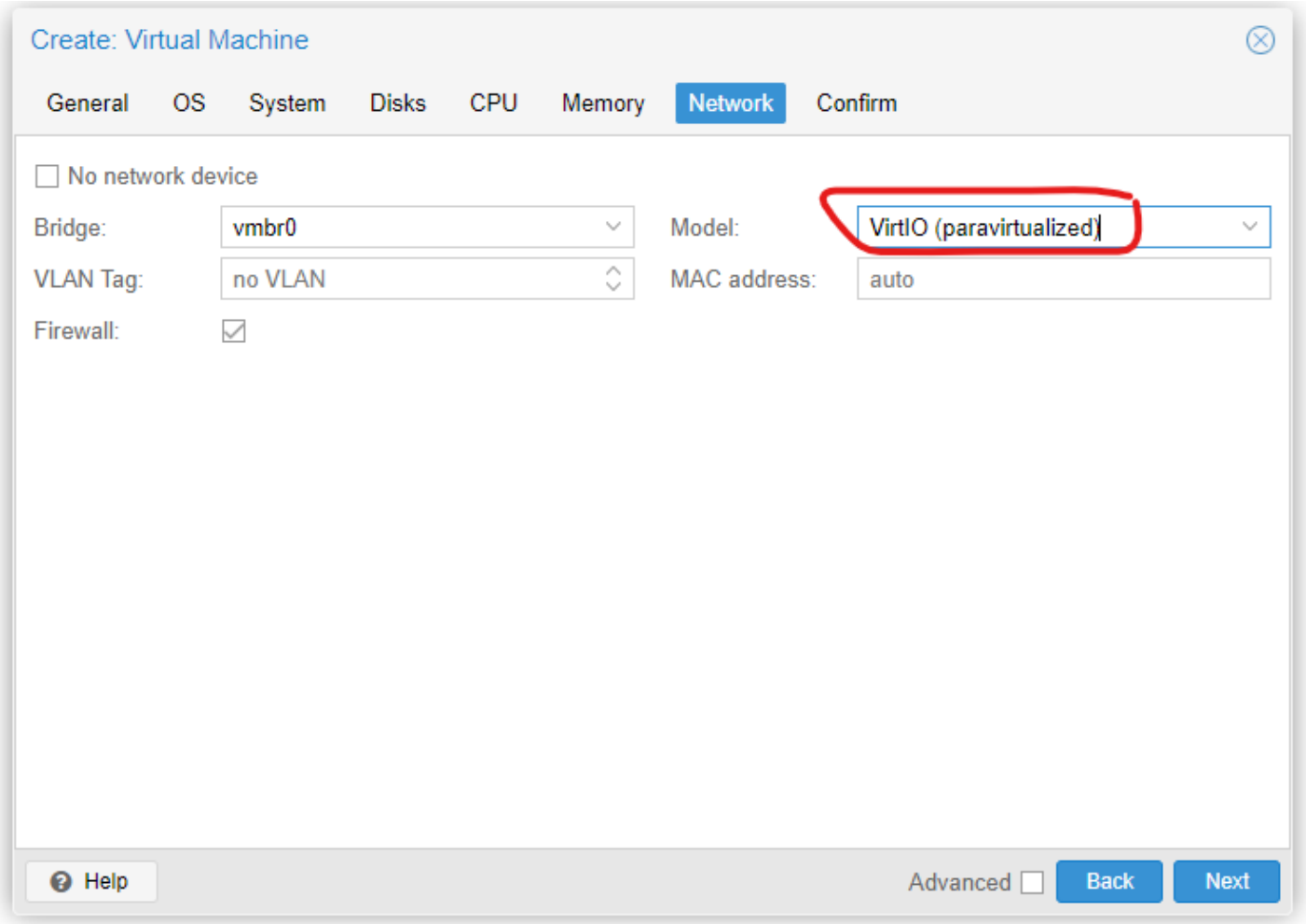



Create: Virtual Machine
General
OS
System
Disks
CPU
Memory
Network
Confirm
Use CD/DVD disc image file (iso)
Storage:
local
ISO image:
win-serv-19-en.iso
Use physical CD/DVD Drive
Do not use any media
Guest OS:
Type:
Microsoft Windows
Version:
10/2016/2019
Advanced
Back
Next

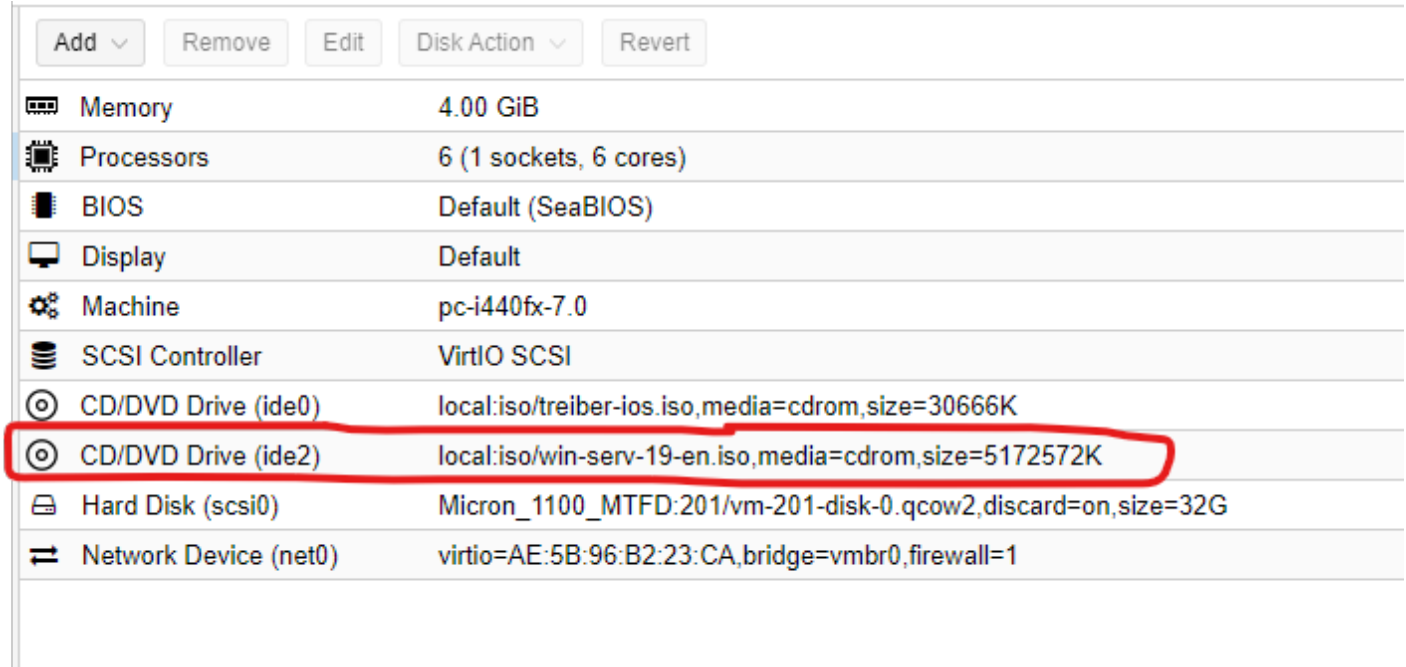

Create: Virtual Machine
General
OS
System
Disks
CPU
Memory
Network
Confirm
Graphic card:
Default
SCSI Controller:
VirtIO SCSI
Machine:
Default (i440fx)
Qemu Agent:
Firmware
BIOS:
Default (SeaBIOS)
Add TPM:
Help
Advanced
Back
Next

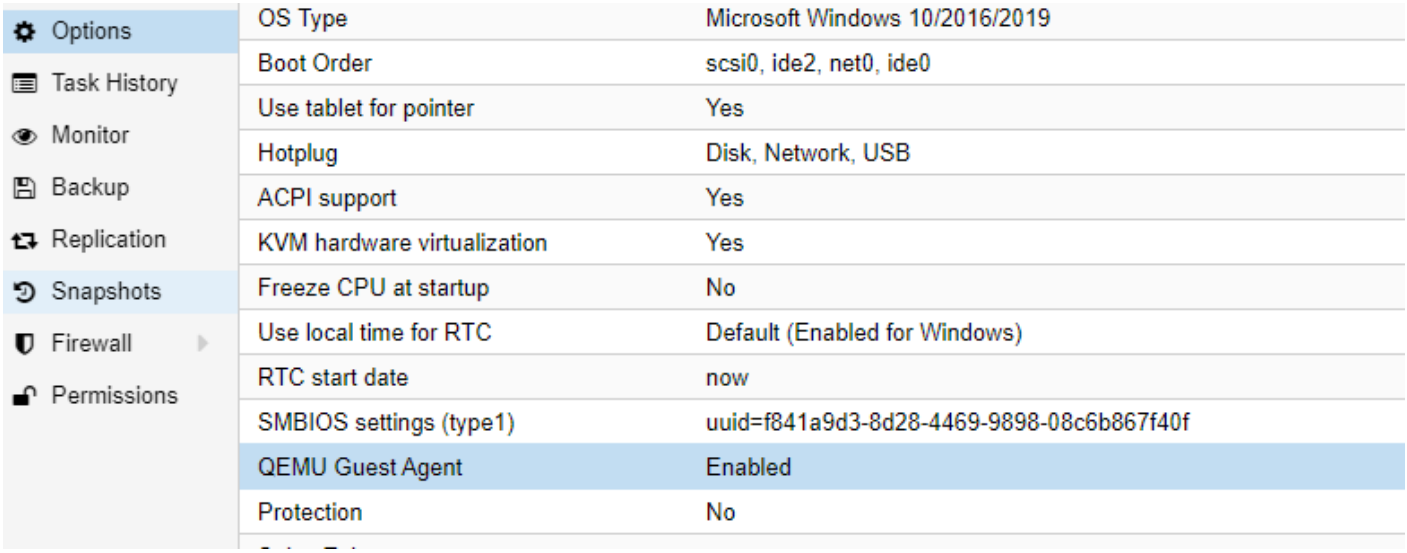


Für beste Performance:

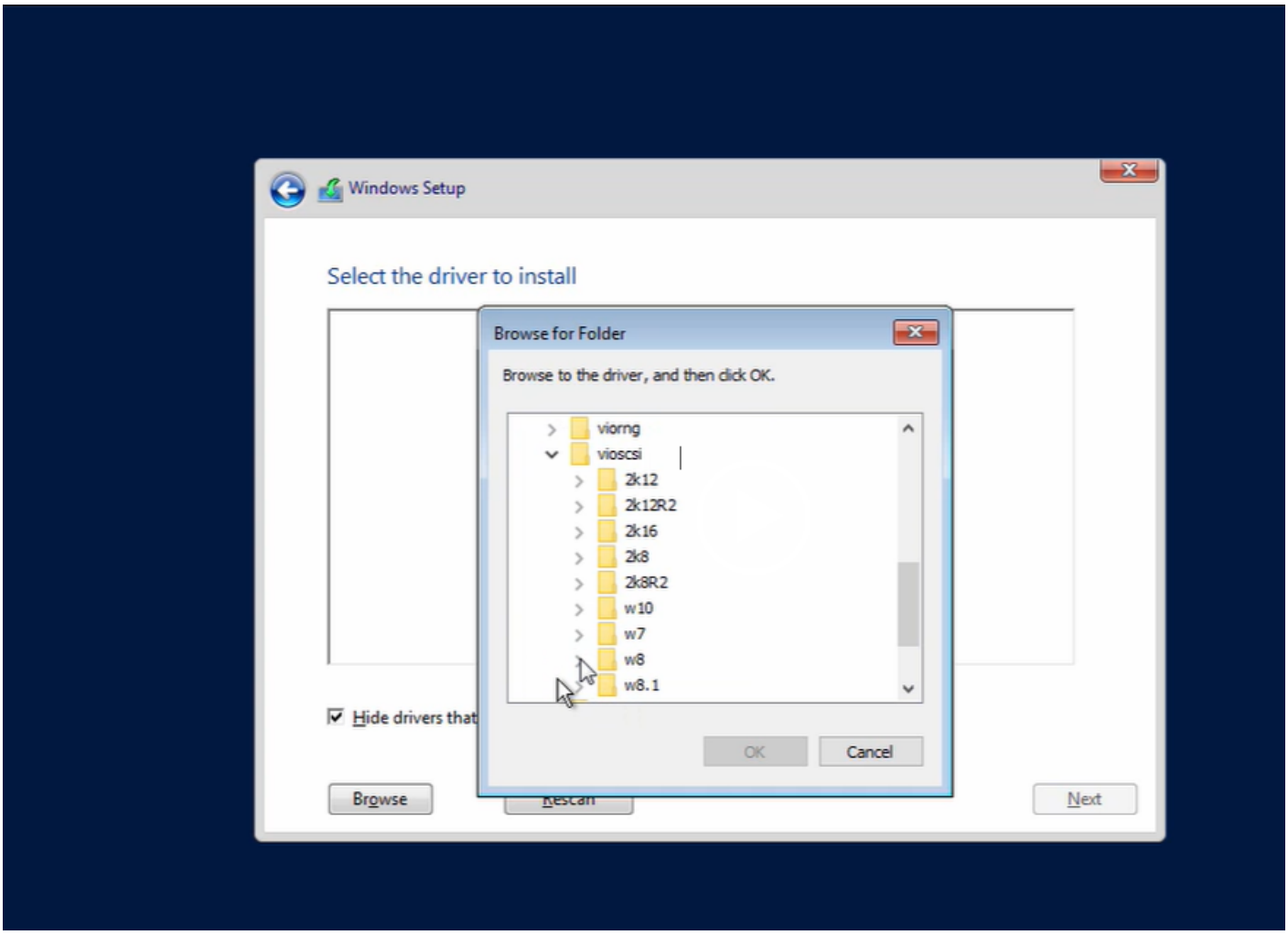


Treiber ISO einlegen

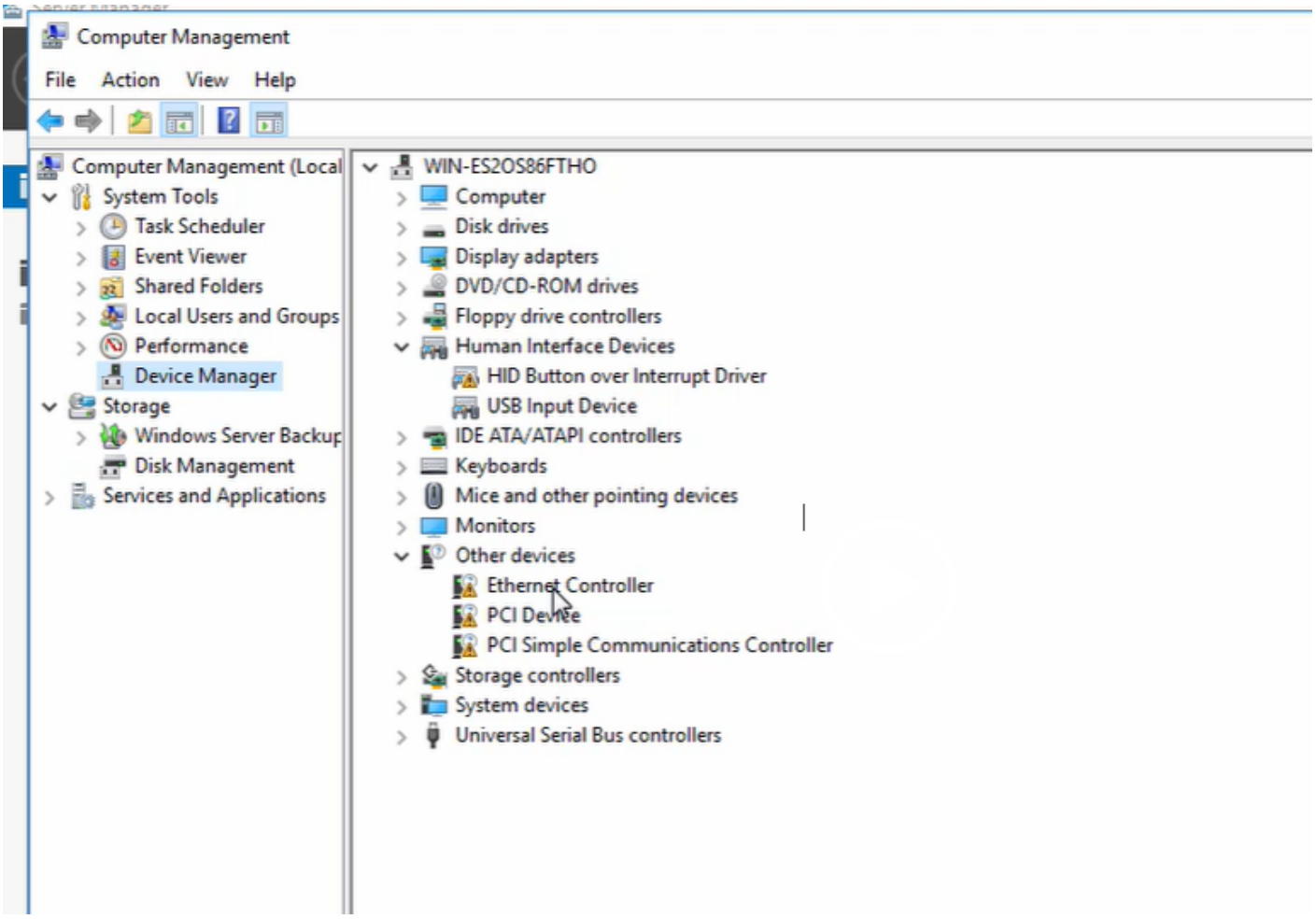


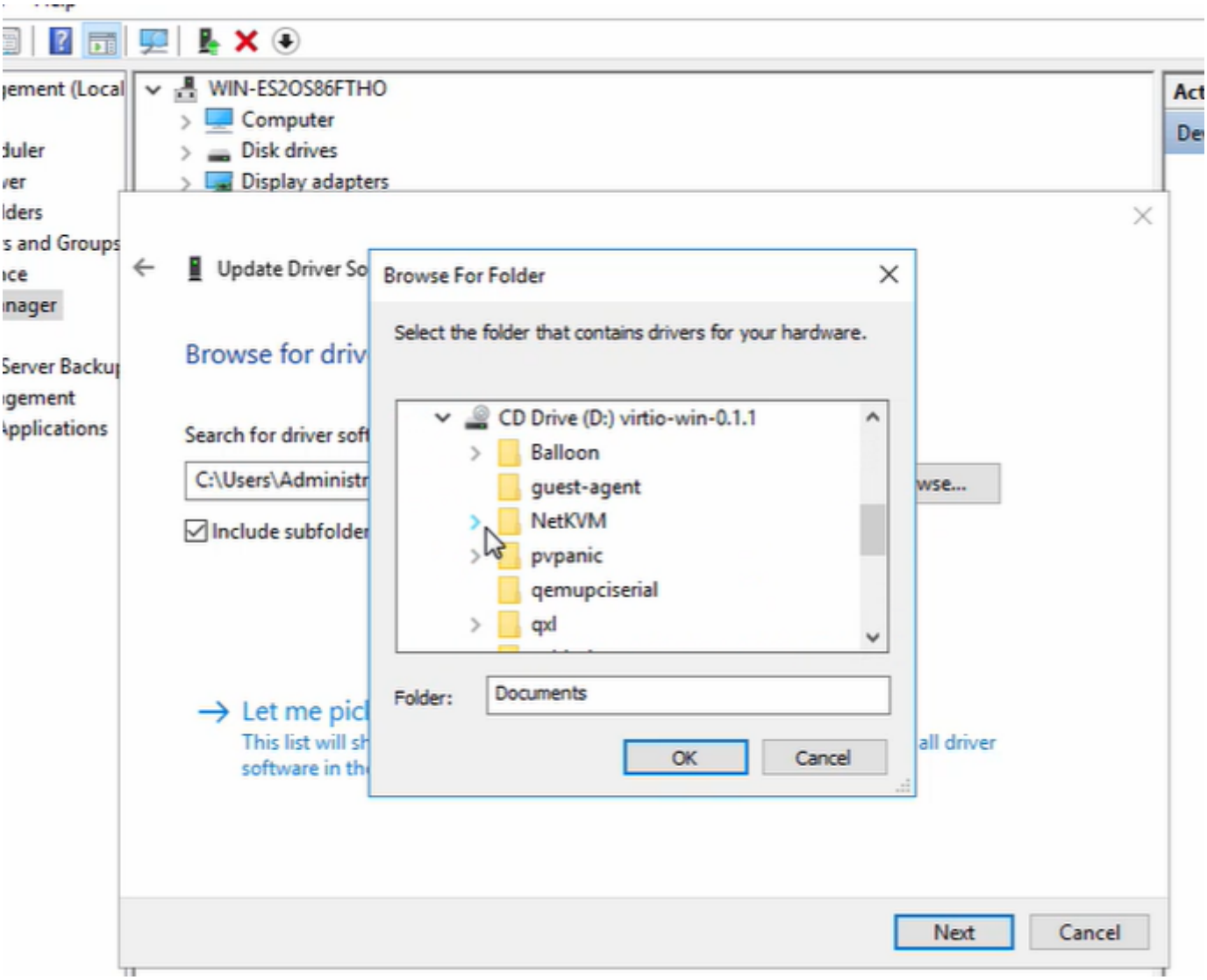


VM Starten und Windows installieren



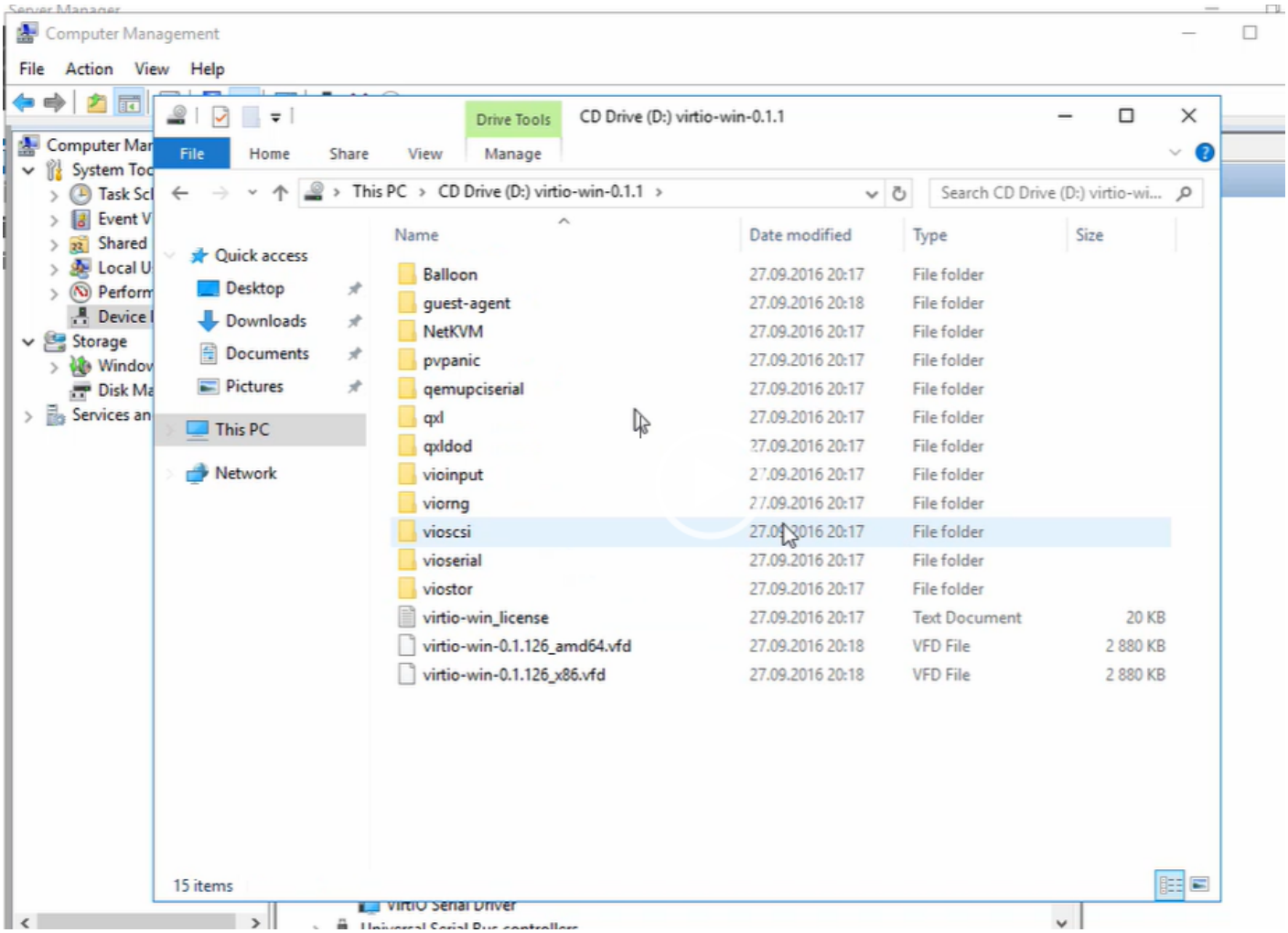
Bei der Installation die wird keine Festplatte angezeigt. Hier die ISO mit den Treibern auswählen und den Treiber VIOSCSI auswählen


Computer Management
File Action View Help
Computer Management (Local
System Tools
Task Scheduler
Event Viewer
Shared Folders
Local Users and Groups
Performance
Device Manager
Storage
Windows Server Backup
Disk Management
Services and Applications
WIN-ES2OS86FTHO
Computer
Disk drives
Display adapters
DVD/CD-ROM drives
Floppy drive controllers
Human Interface Devices
HID Button over Interrupt Driver
USB Input Device
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Monitors
Other devices
Ethernet Controller
PCI Device
PCI Simple Communications Controller
Storage controllers
System devices
Universal Serial Bus controllers

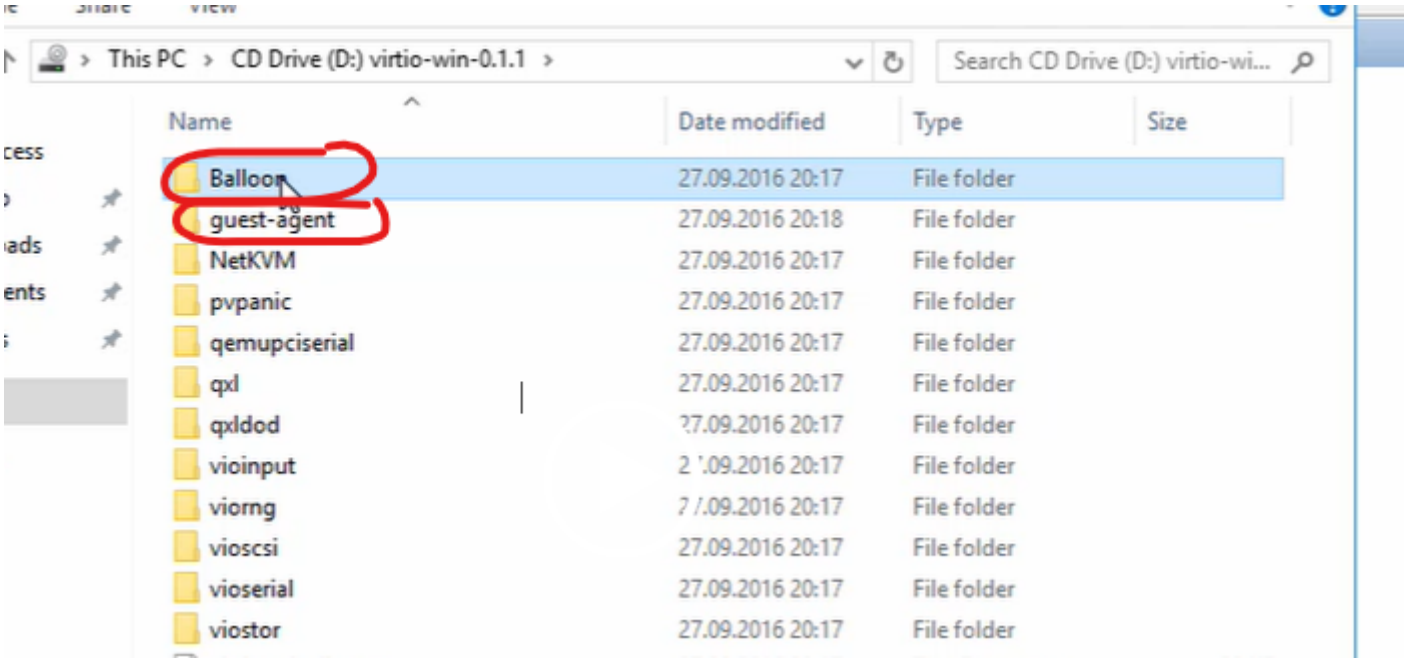


Auch mit PCI Devices machen: Nur DVD auswählen (E:) und Windows sucht den Treiber selbstständig

PCI Simple genau so



QEMO Guest Agent installieren



Balloon installieren:

Ballon amd64 Ordner in C:Programme kopieren und in Balloon umbenennen

CMD > Navigiere zu C:Programm/Ballon und `blnsvr.exe -i` eingeben

Treiber Installieren in der Konsole

```
pnputil -i -a Treiber_inf_Datei.inf
```

Proxmox Terminal

Befehle

`qm list`	Alle VMs anzeigen
`qm list --state running`	Alle laufenden VMs anzeigen
`qm status <VMID>`	Weitere Informationen zur VM
`pct list`	Alle Container anzeigen
`pct list --status running`	Laufende Container
`pct status <CTID>`	Weitere Informationen zum Container
pct enter <id>	Damit gelangt man direkt in die CT
pct start <id>	Contianer Starten
pct stop <id>	Container Stoppen

Container mit einem Befehl erstellen:

```
pct create 100 local:vztmpl/ubuntu-24.04-standard_24.04-1_amd64.tar.zst \
  --hostname openwebui \
  --memory 8 \
  --cores 4 \
  --rootfs local-lvm:32 \
  --net0 name=eth0,bridge=vmbr0,ip=dhcp \
  --features nesting=1
```

USB an VM verbinden

Verbinden

Herausfinden welchen USB-Stick angesteckt ist

```
lsusb
```

Zuweisen

```
qm set 306 -usb0 host=090c:1000
```

Wobei 306 die Nummer der VM ist

Trennen

```
qm set 307 -delete 'usb0'
```

VM mit Cronjob starten

Um eine VM mit Cronjob zu starten muss man den qm befehl direkt angeben.

In der Tabelle crontab -e folgendes eingeben:

`00 12 * * * /usr/sbin/qm start 107` Damit startet man VM Nr. 107

Installieren von Raspiblitz auf Proxmox

Install Raspiblitz on Proxmox

Here I want to show you how to install a new Raspiblitz on a Debian VM on Proxmox and get it running. My Raspiblitz ran very long and stable on a Raspberry Pi 4 with 8GB RAM. It would very likely continue to do so for a longer time, however my Lightning Node is growing more and more and various apps and services are built on top of my Node. So the issue of availability and backup becomes more and more important. Therefore I decided to migrate the Raspiblitz to a VM in Proxmox. So I have much more room to maneuver regarding backup and administration.

This guide here will help you to set up a completely new Raspiblitz with Proxmox. The guide for the migration will follow soon…

What is needed?

- Proxmox installation on an Intel NUC, laptop or server
- at least 1TB SSD

You have several options for the SSD: Either you install the 1TB SSD in the system and install your Proxmox host on it or (as I did) you have an internal SSD (in my case 500GB M2 SSD) where the host operating system is located. I connected the 1TB SSD via SATA to my Intel NUC. This is used exclusively for storing the blockchain and Lightning Node.

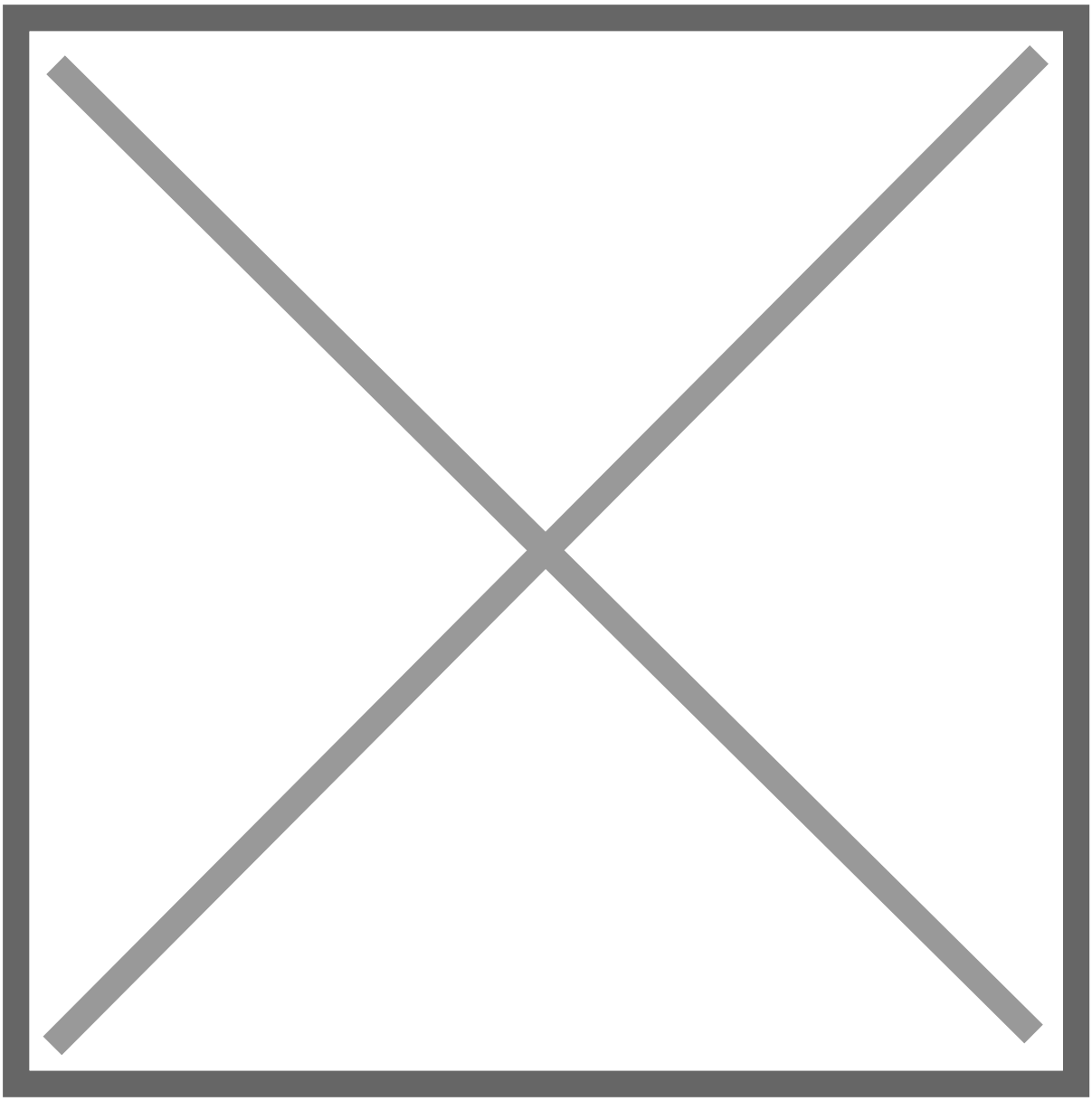
Create Debian VM

We install Raspiblitz on a fresh Debian machine. Therefore we have to download the ISO file from Debian first. Here just choose the right processor architecture: (For me it is amd64)

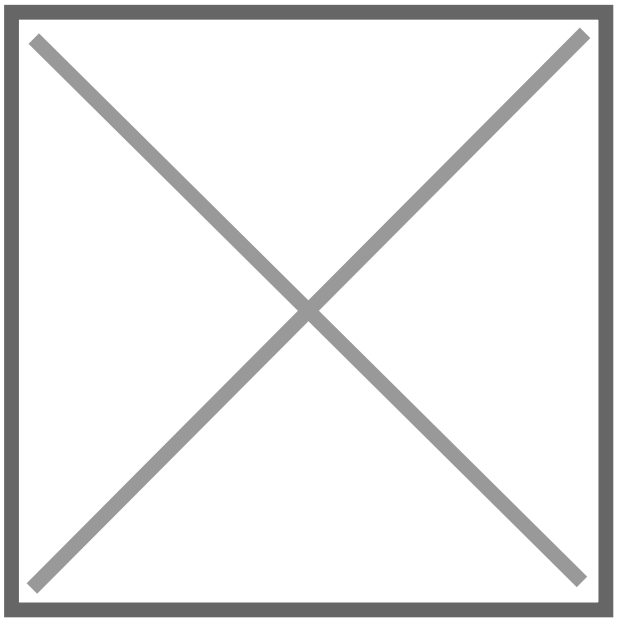
https://www.debian.org/distrib/

https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-12.5.0-amd64-netinst.iso

Afterwards this ISO file can be uploaded under Proxmox. To do this, click on the Local Storage, ISO Images and on Upload:

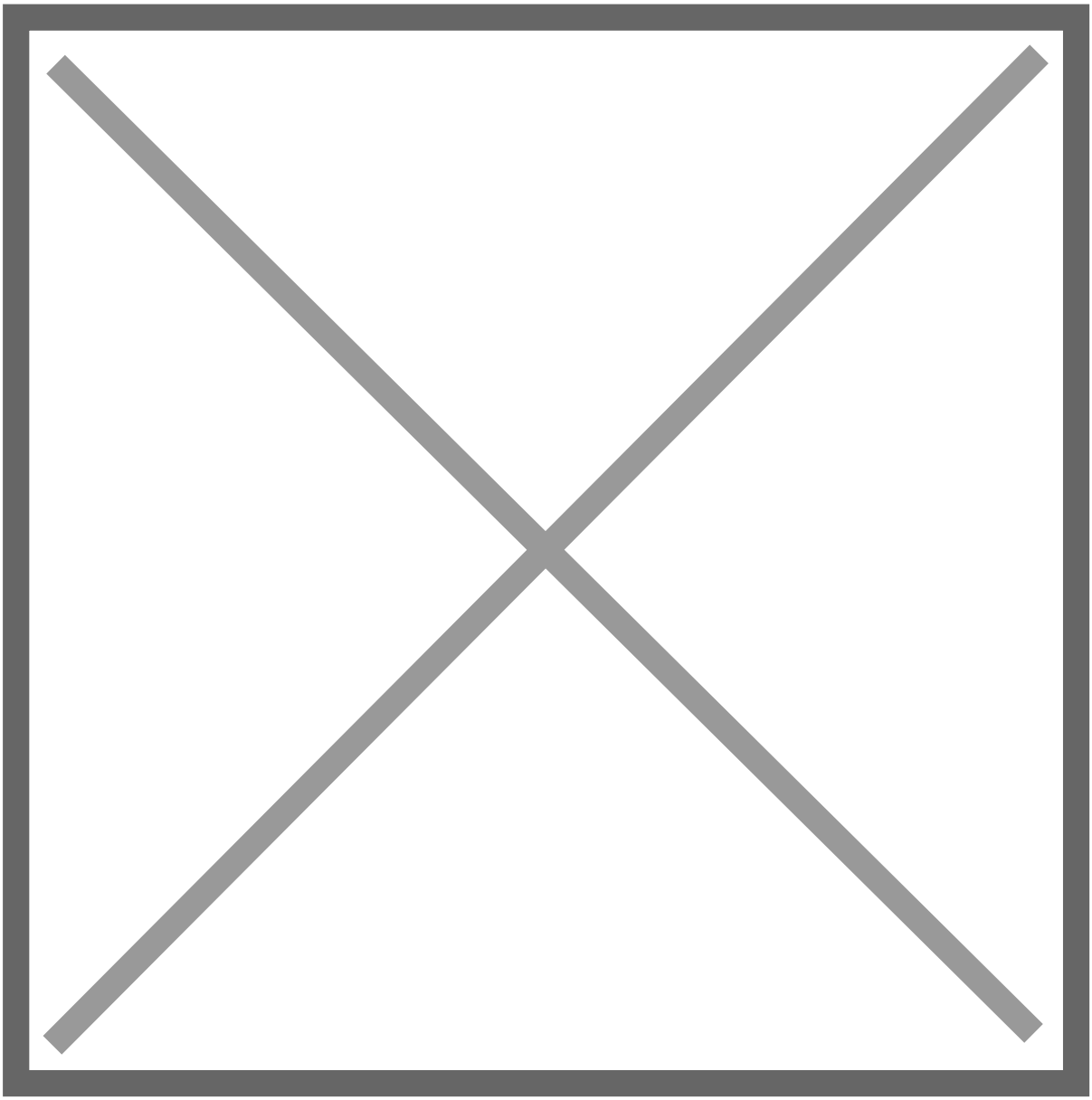


Now you can create a new virtual machine by clicking on “Create VM” in the upper right corner. Now we click on it.

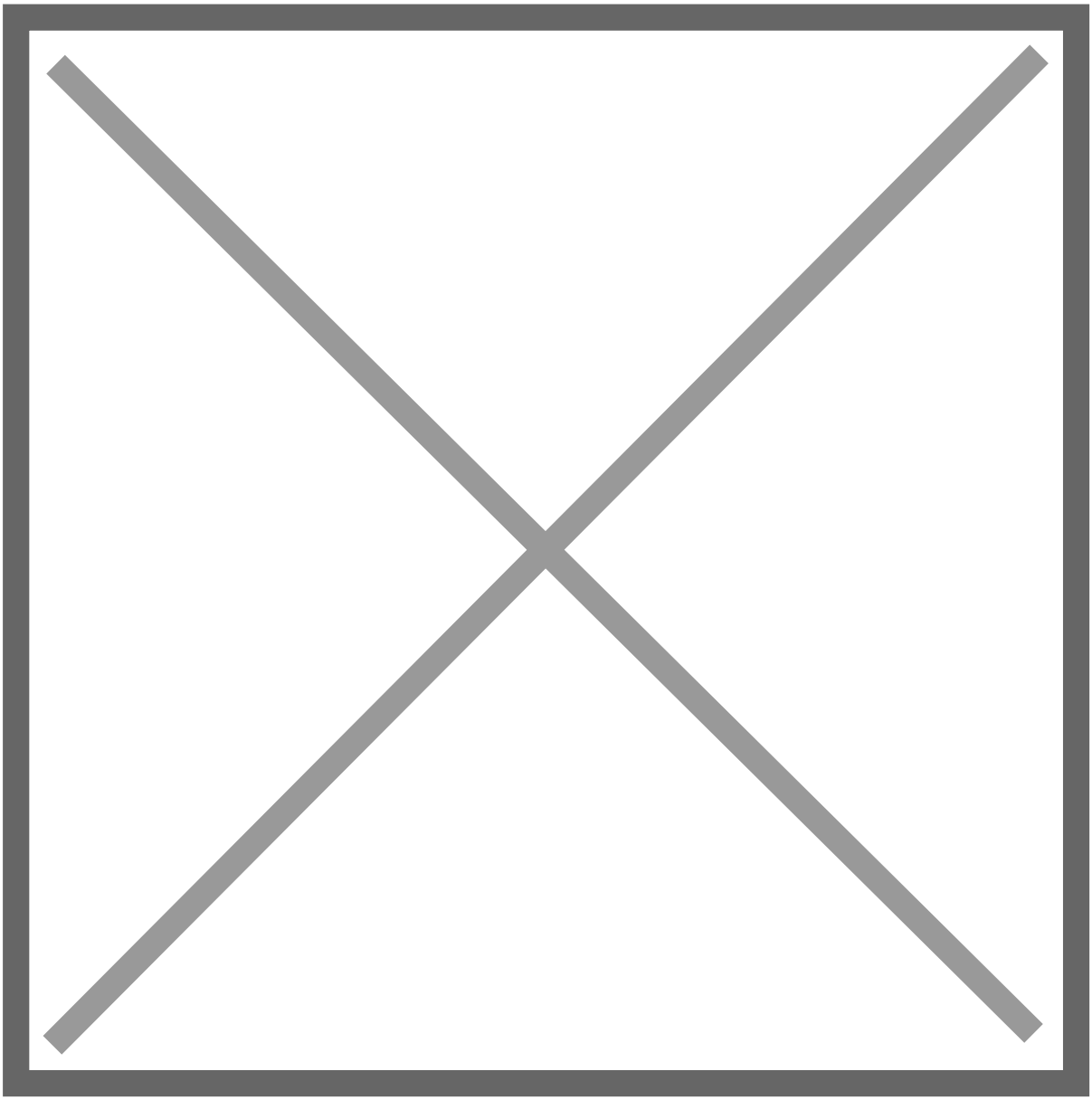


A settings window appears where we can now specify the properties of the VM. In my example, the properties look like this:

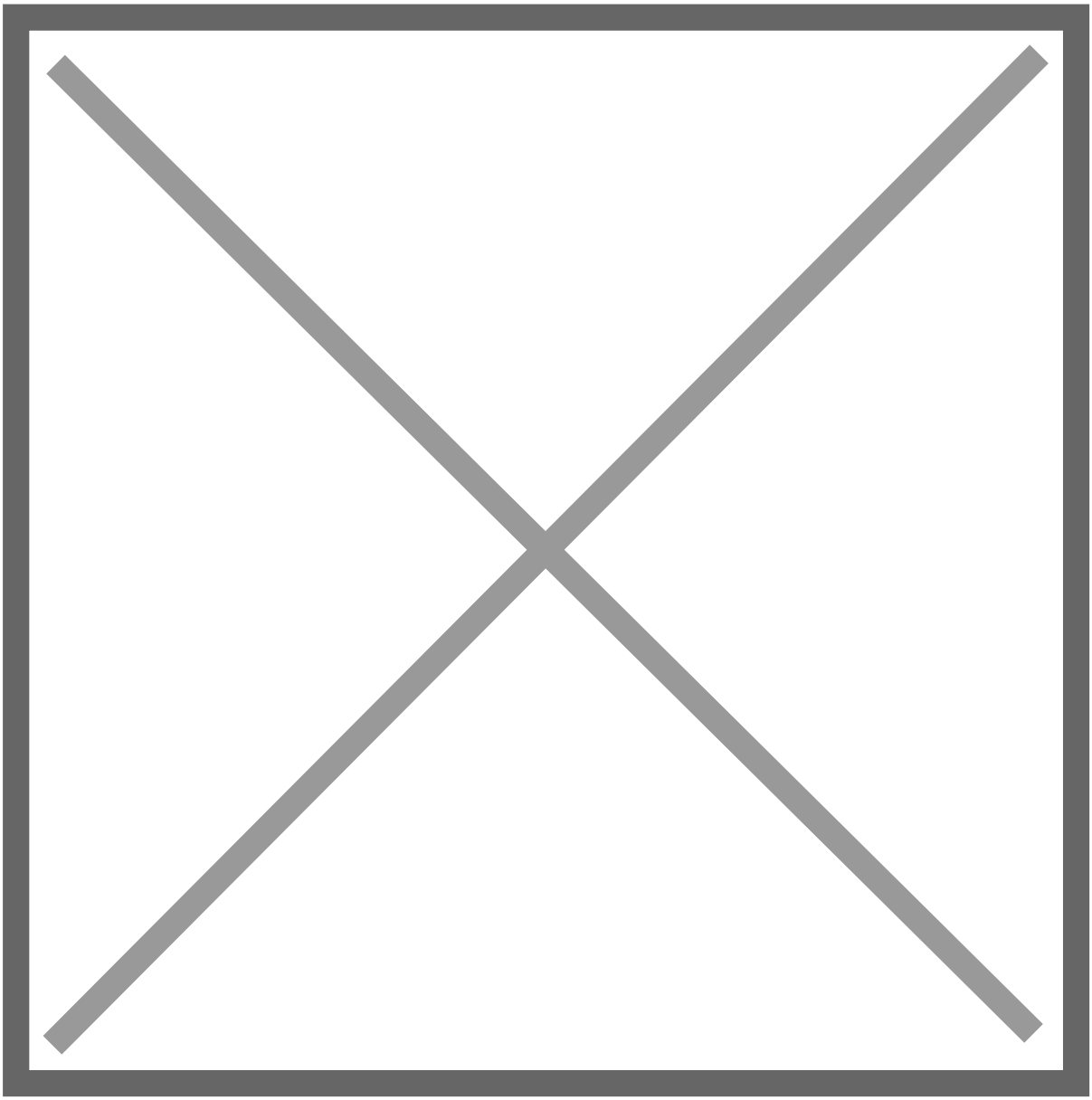
VM ID and name can be selected by yourself.



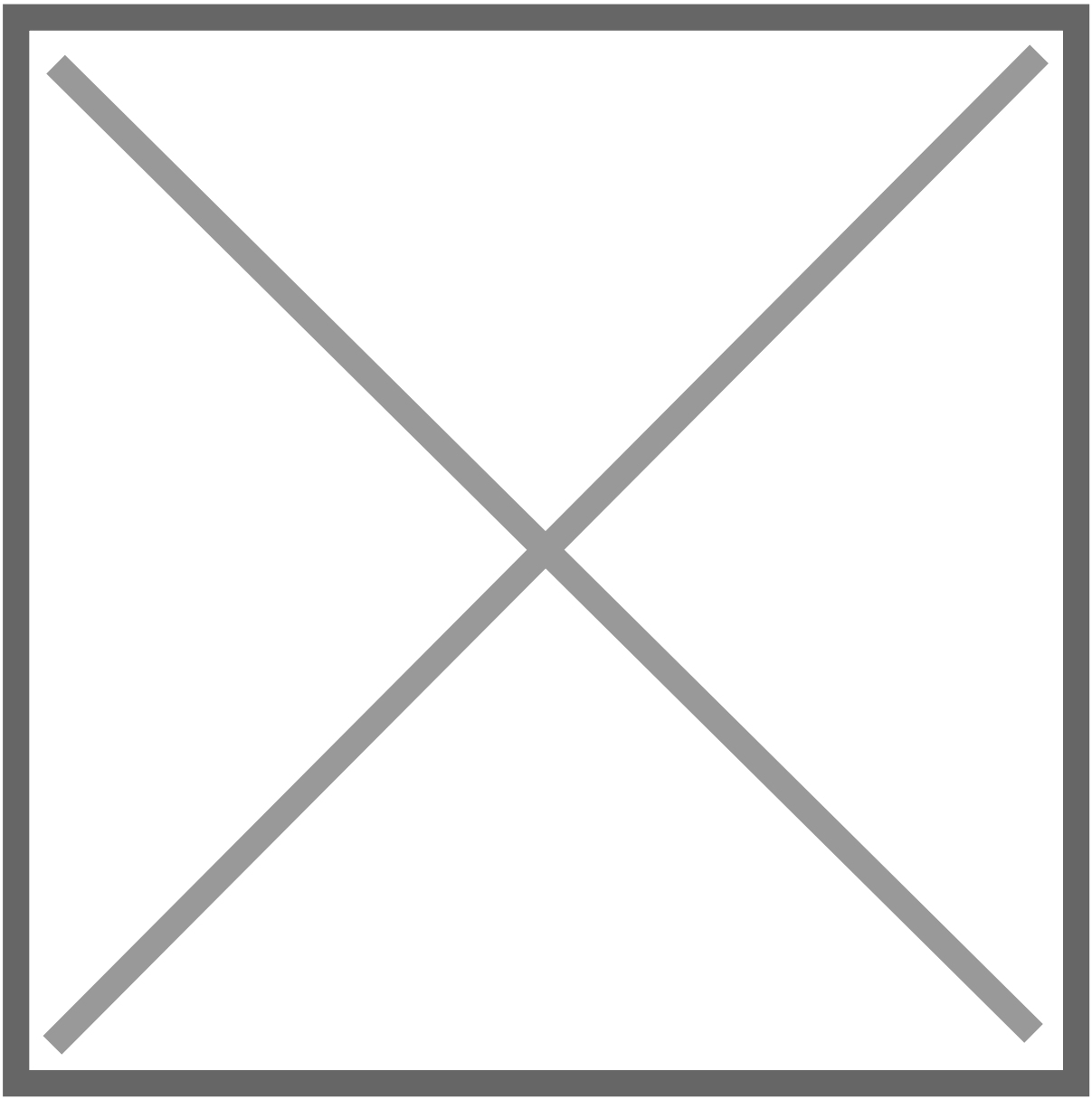
Under the item “OS” we now select the previously downloaded ISO file:



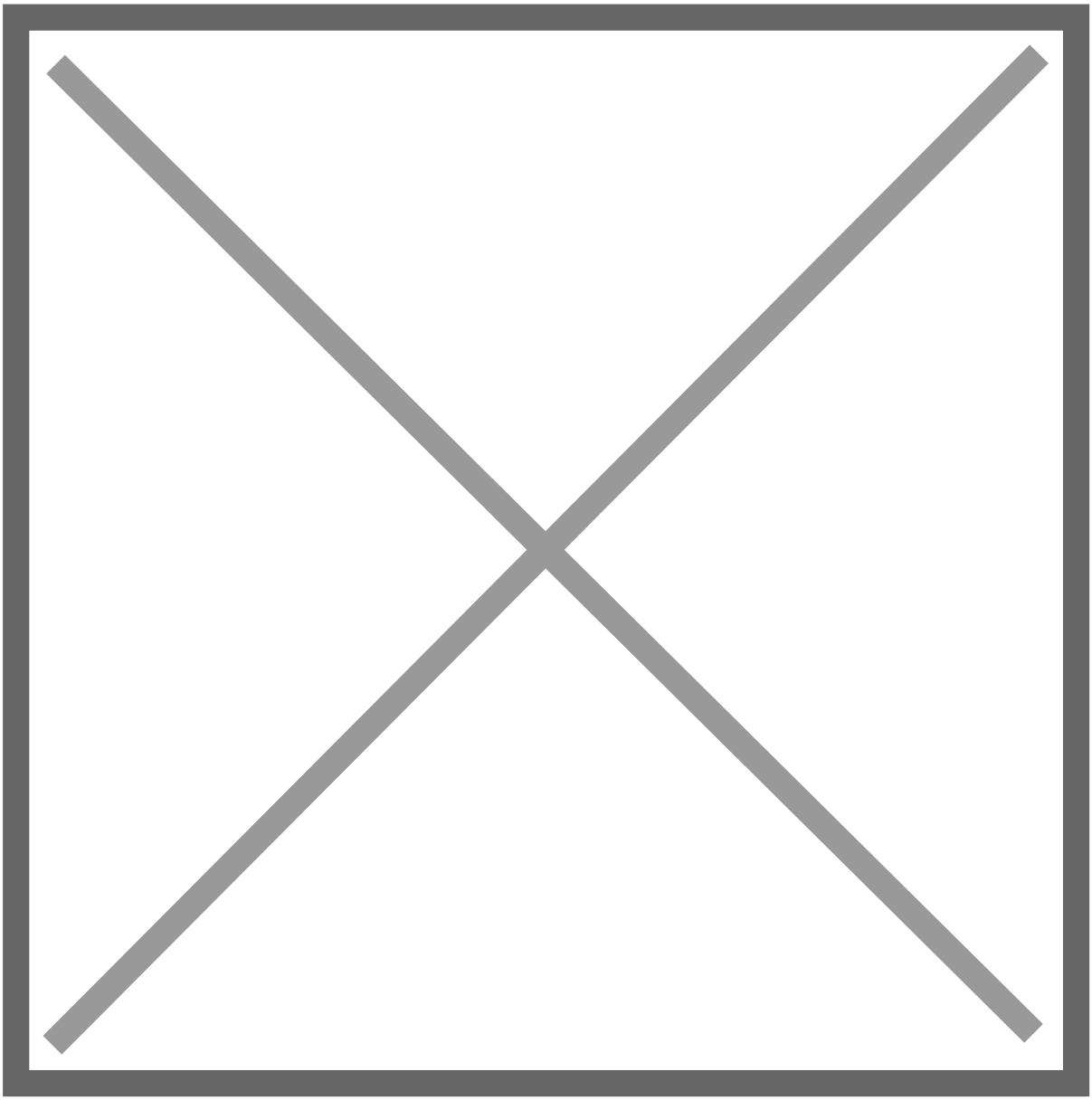
Under “System” we can leave the default settings. I have checked Qemu Agent so that the VM can communicate with the Proxmox host via Qemu Agent and transfer data.



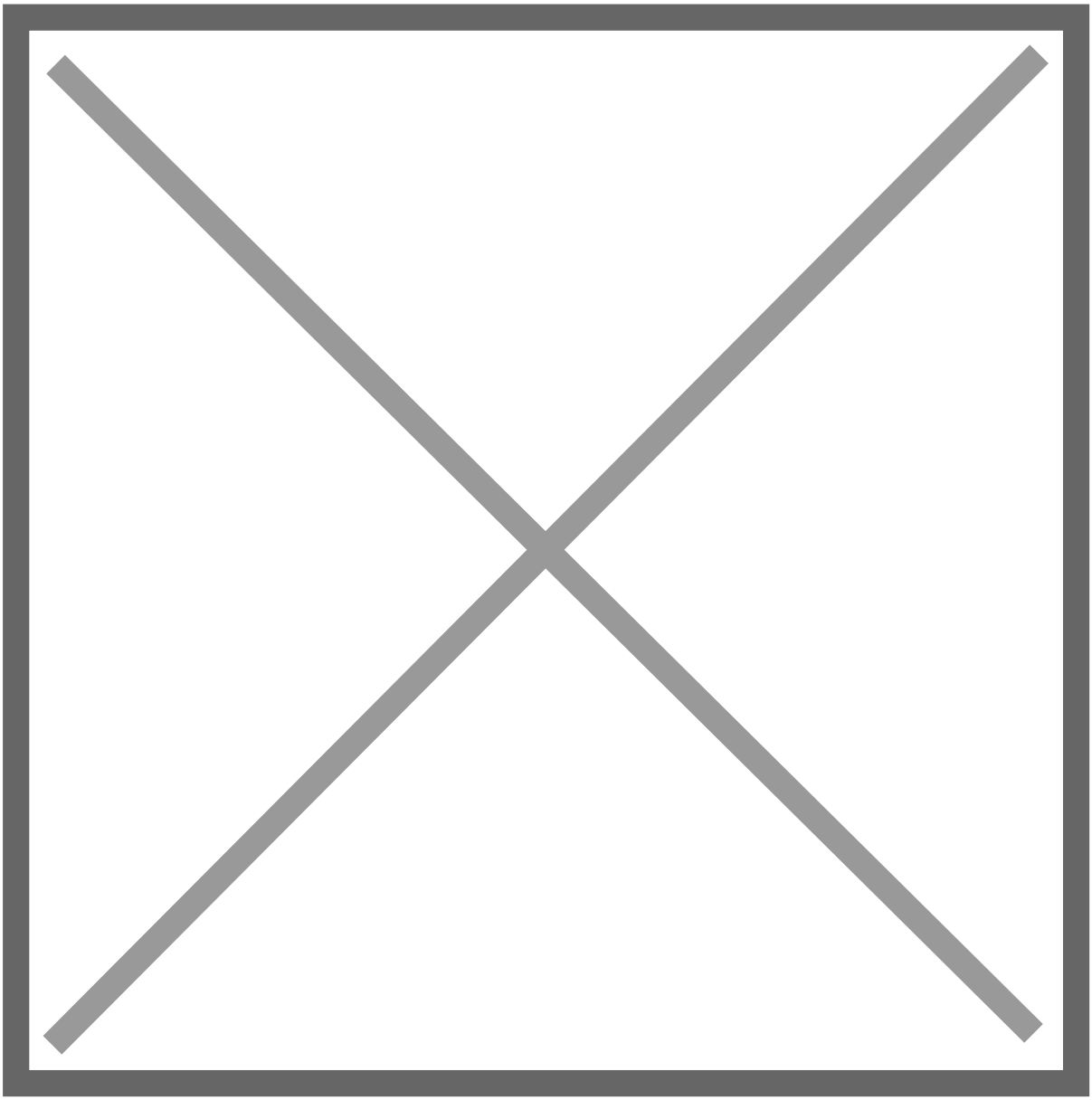
Under “Disks” you can now specify the desired size of the VM. I have set the same size (32GB) as my SD cards from the Raspiblitz are big. This can be increased at any time in the future, if you need more space and the host machine also has this space available. Another advantage why I virtualized my Raspiblitz □□



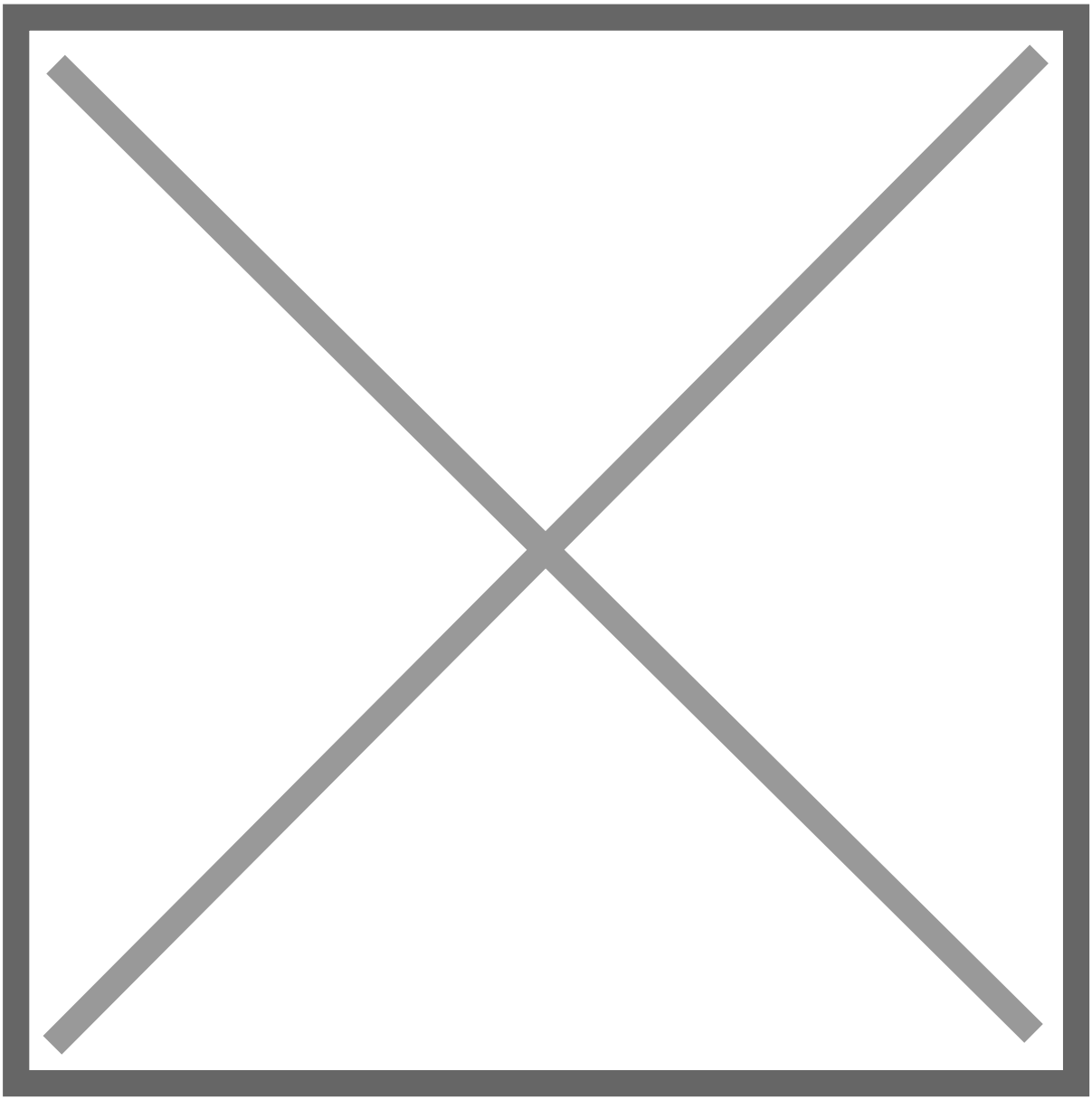
Under “CPU” you can specify the desired number of cores. Of course, this depends on your host operating system. My Intel NUC has 4 cores, so I can provide 4 cores to the VM.



Under “Memory” you must specify the desired RAM number in MB. This also depends on your host. If possible, I would enter 8GB or more. **Small hint: 1GB = 1024 MB. So 8GB = 8192 MB. (8 x 1024)**



The VM also needs a network adapter. You also have to select this adapter based on your Proxmox installation. Default will be vmbr0. On my Proxmox I have configured several VLAN, where vmbr3 is my BTC VLAN. But this is only the case for me.



After that you can click on “Finish” and the VM will be created. This now also appears with name on the left side and can now be started. (Right click -> Start)

Now you can open the console (top right) of the VM and do the Debian installation steps normally. I have abbreviated the steps a bit here:

- Install
- Select Language
- Select Location
- Select Keyboard Language
- Set Hostname
- Set domain (or just leave it empty)
- Set root password
- Create new user (mine is called “pi”)
- Set password for user pi

- Guided – use entire disk
- Select SCSI3 harddisk
- All files in one partition
- Finish partitioning and write changes to disk
- “Write the changes to disks?” -> Yes
- Scan extra installation media? -> No
- Package manager -> Select your country
- Package manager -> deb.debian.org
- http proxy -> leave empty and continue
- Participate in the package usage survey? -> No
- Software selection: SSH server and standard utilities should be sufficient here
- Install the GRUB boot loader to your primary drive? -> Yes
- Select /dev/sda

The VM is now installed and starts to boot. In the meantime you can remove the ISO file. (VM -> Hardware -> CD/DVD Drive -> Do not use any media -> OK)

Add storage

Now to install Raspiblitz on this new VM we need to do 2 things: Connect the hard disk for the blockchain data and pass it to the VM and install the SD Card Builder Script of Raspiblitz. We start first with the hard disk, where there are 2 variants:

Variant 1: External hard disk

Now connect the hard drive to the host system using SATA or USB. In my example here I use a USB hard disk, which I connected via USB 3.1 to the Intel NUC. The best thing to do now is to shutdown the Raspiblitz VM.

Now you need to log in to the console on the Proxmox host system and do the following:

https://www.youtube.com/embed/U-UTMuhmC1U

The commands from the video here again to copy:

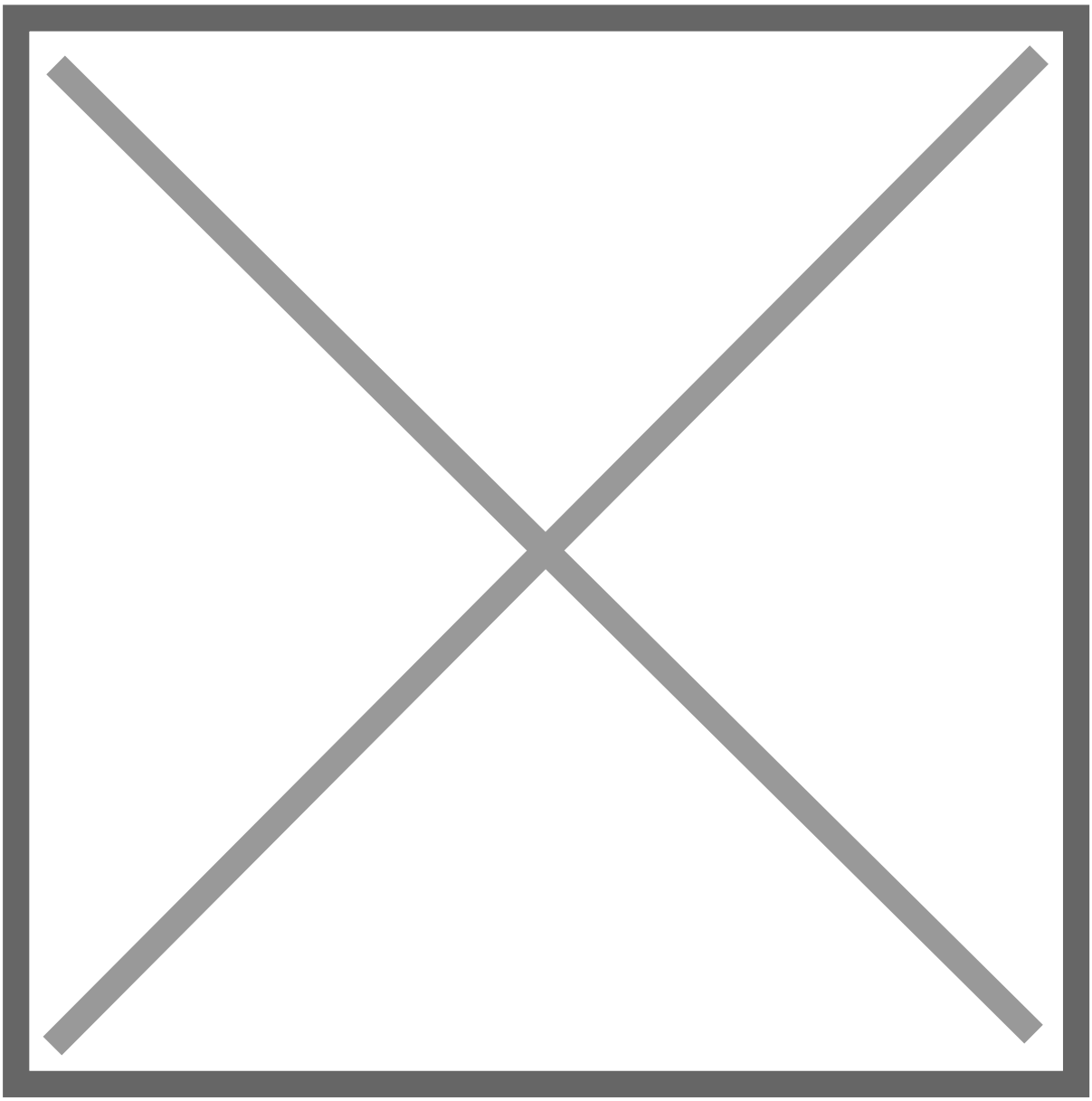
```
ls -n /dev/disk/by-id/
/sbin/qm set [VM-ID] -virtio2 /dev/disk/by-id/[DISK-ID]
```

It is important here that the hard disk is passed through by ID. If something changes in the dev sda order in the future, the correct hard disk is still connected to the VM.

Variant 2: Use internal storage

If you have enough space on the host operating system, you do not have to use an external hard disk. You can simply add a second hard disk to the VM under “Hardware -> Add -> Hard Disk”. I would recommend at least 1TB as storage size.

Regardless of whether variant 1 or 2 was executed, the VM should now have 2 hard disks connected in the hardware overview: A smaller one (e.g. 32GB) where the operating system of Raspiblitz will be installed and run and a larger one (e.g. 1TB or more) where all the blockchain data will be stored later.



Install Raspiblitz

Now we are ready to install Raspiblitz via script. For this we start the Raspiblitz VM and log in as root user in the console. First of all update everything:

```
apt update
apt upgrade -y
apt install sudo
```

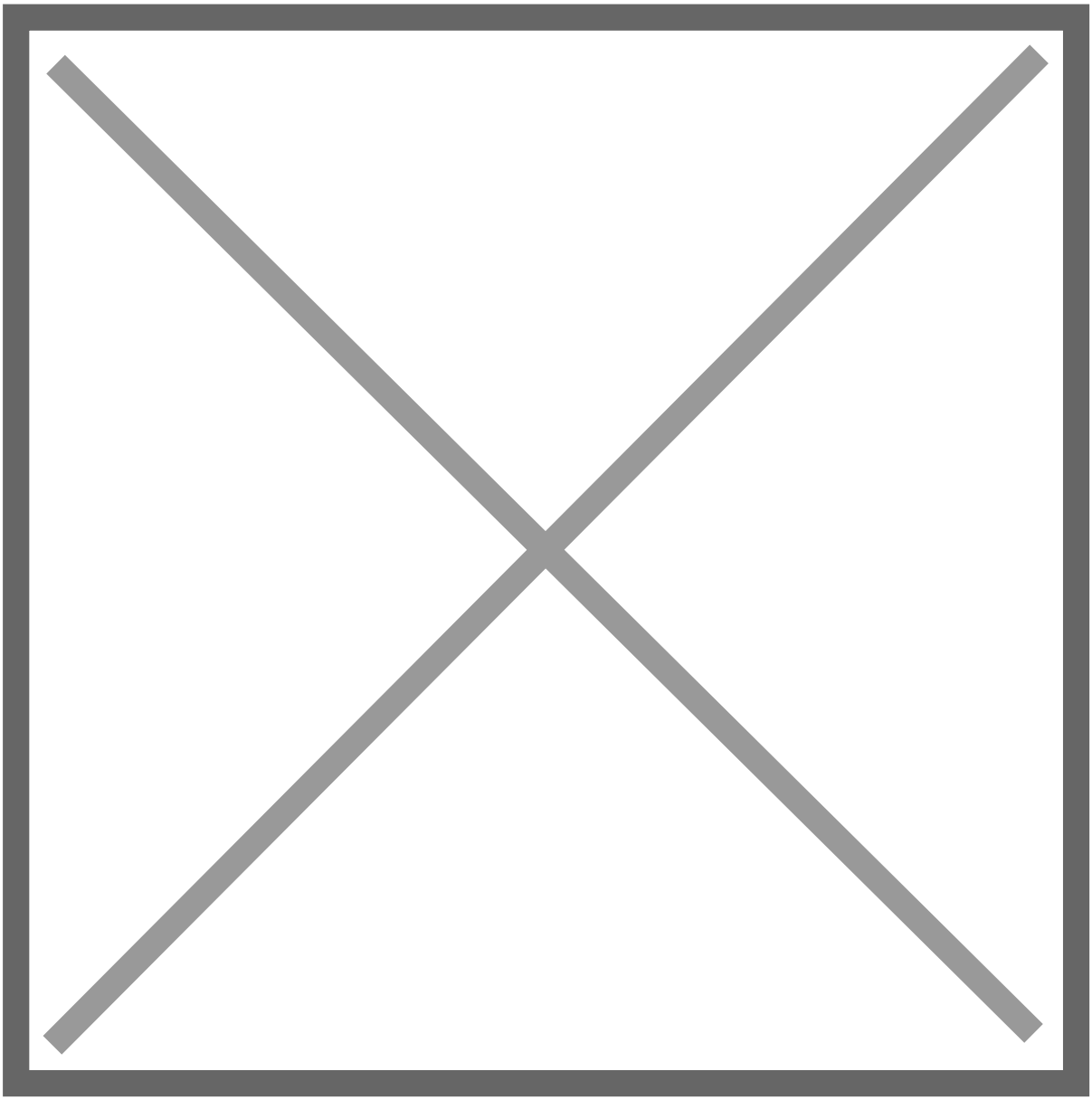
Now we need to download the Build SDCard Script from Rootzoll. The version can be customized as you like. The latest version is the 1.11.

```
wget https://raw.githubusercontent.com/raspiblitz/raspiblitz/v1.11/build_sdcard.sh
```

And run:

```
sudo bash build_sdcard.sh -f false -d headless -t false -w off
```

The script now shows you information from your system. If all this is correct, start with “yes”.

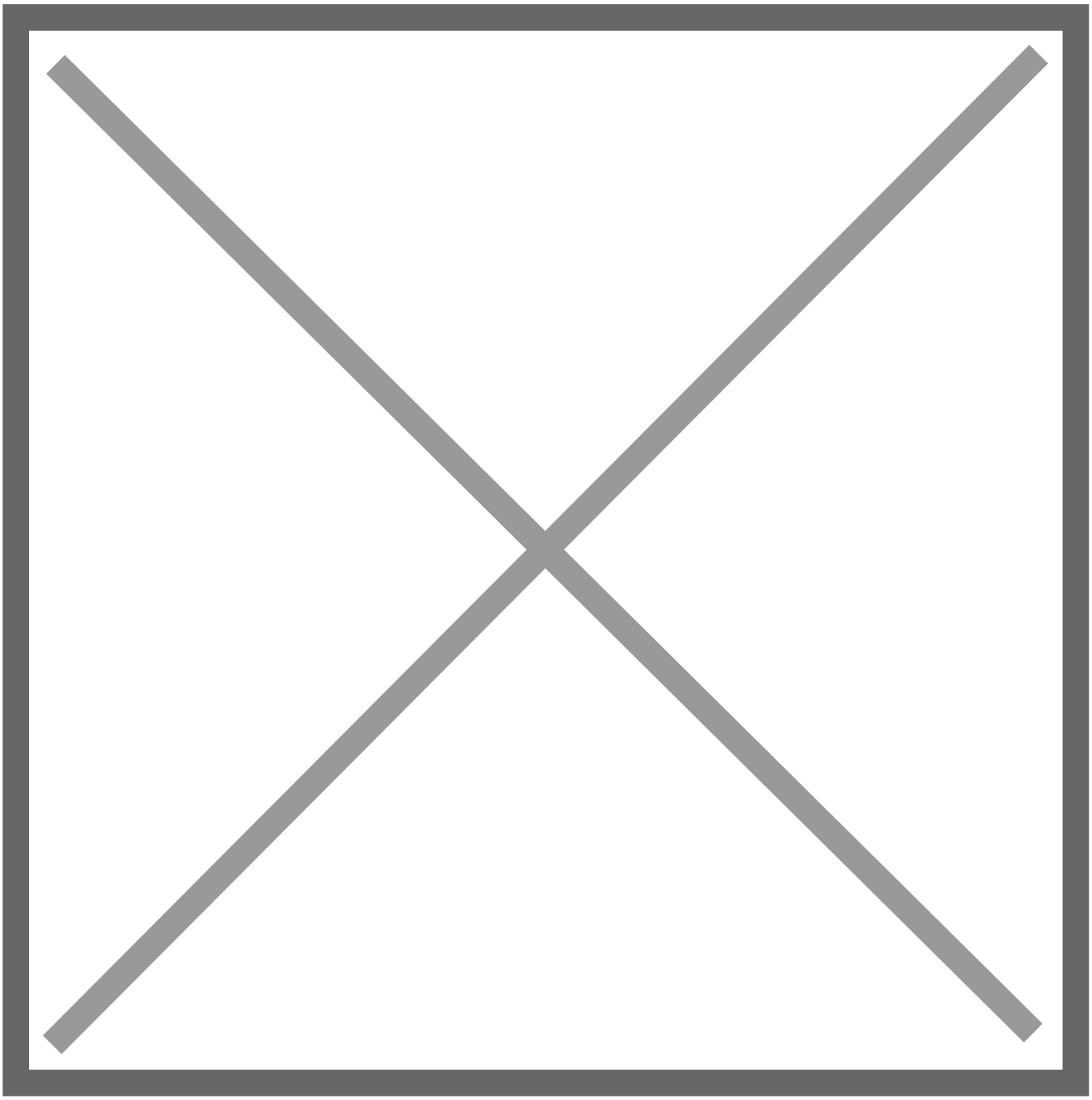


Now the installation takes a few minutes. Do not abort or shut down the VM here, just let it install. When everything is finished, a reboot is needed:

```
sudo shutdown -r now
```

Now you can call the IP address of your VM in the browser and perform the normal installation steps of Raspiblitz.

Update Raspiblitz auf VM



May 5, 2024

Raspiblitz auf Proxmox updaten

Der Update Prozess mit einem Raspiblitz virtualisiert auf Proxmox funktioniert im Grunde genau gleich wie mit einem Raspiblitz auf einem Raspberry Pi. Der Prozess da sieht ja wie folgt aus: Im Raspiblitz Menü auf Relase Update klicken, Raspi herunterfahren, SD Karte mit neuem Image flashen und der Blitz nun wieder hochfahren. Dieser Prozess kann auch im virtualisierten Setup adaptiert werden. Folgende Schritte beschreiben den Updateprozess, wie ich ihn mache. Beachte hier: Es gibt hier mehrere Möglichkeiten, wie du dein Raspiblitz auf Proxmox updaten kannst.

- Parallel eine neue frische debian VM erstellen
- Raspiblitz mit der neusten Version dort darauf installieren
- Neue VM herunterfahren
- Im Menü beim “alten” Raspiblitz auf updaten klicken
- “Alte” VM auch herunterfahren
- Festplatte mit Blockchain etc. von der alten in die neue VM einhängen
- Neue VM hochfahren

Neue VM installieren

Die parallel aufgesetzte VM kann genau gleich wie in der Anleitung HIER gemacht werden. Wichtig zu beachten ist: Füge die externe Festplatte erst nach der Installation hinzu. Installiere also zuerst einmal Debian (Download Link), melde dich via ssh mit dem neu erstellten Benutzer an und führe folgende Befehle aus:

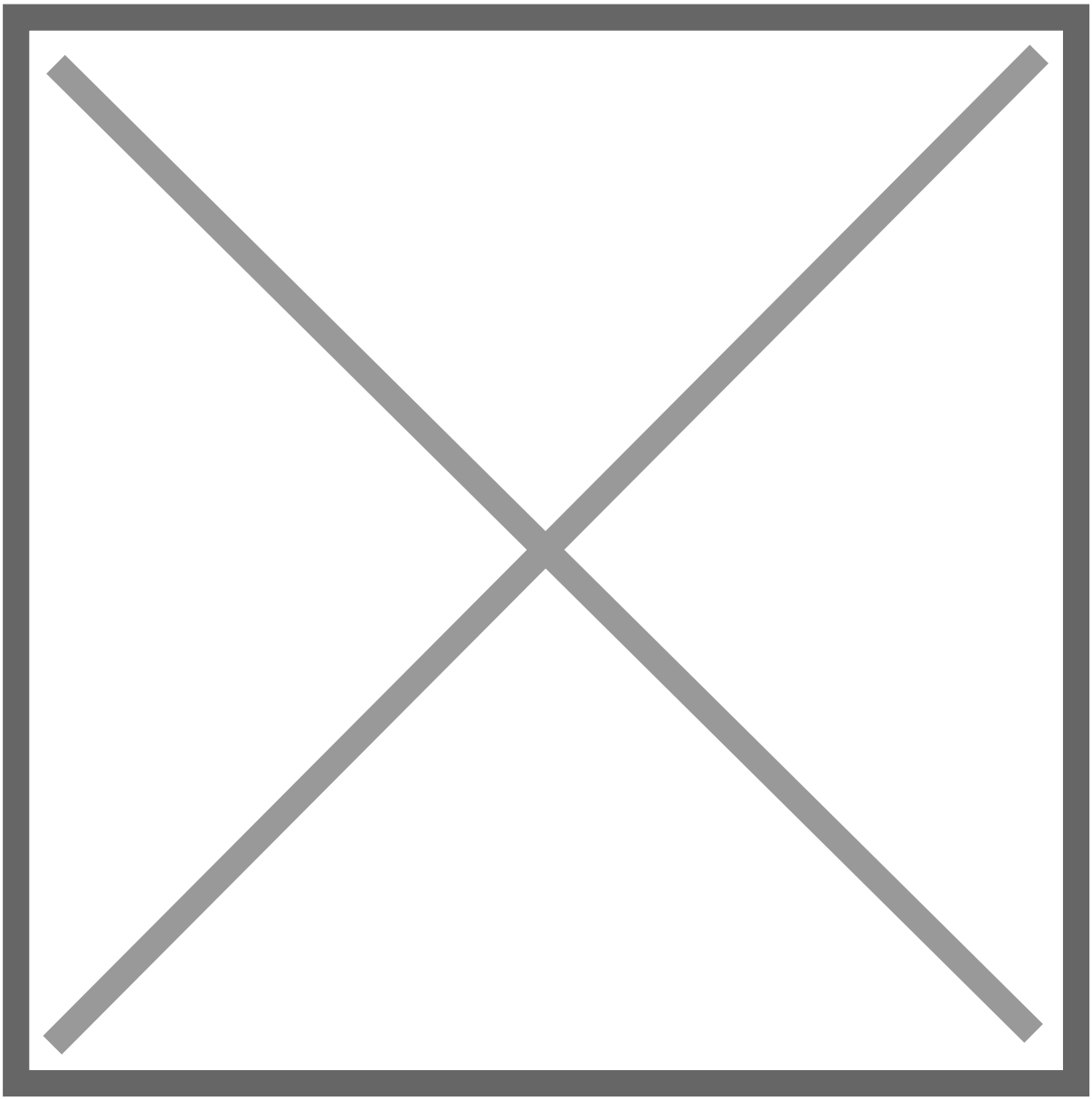
```
wget https://raw.githubusercontent.com/raspiblitz/raspiblitz/v1.11/build_sdcard.sh


sudo bash build_sdcard.sh -f false -d headless -t false -w off
```

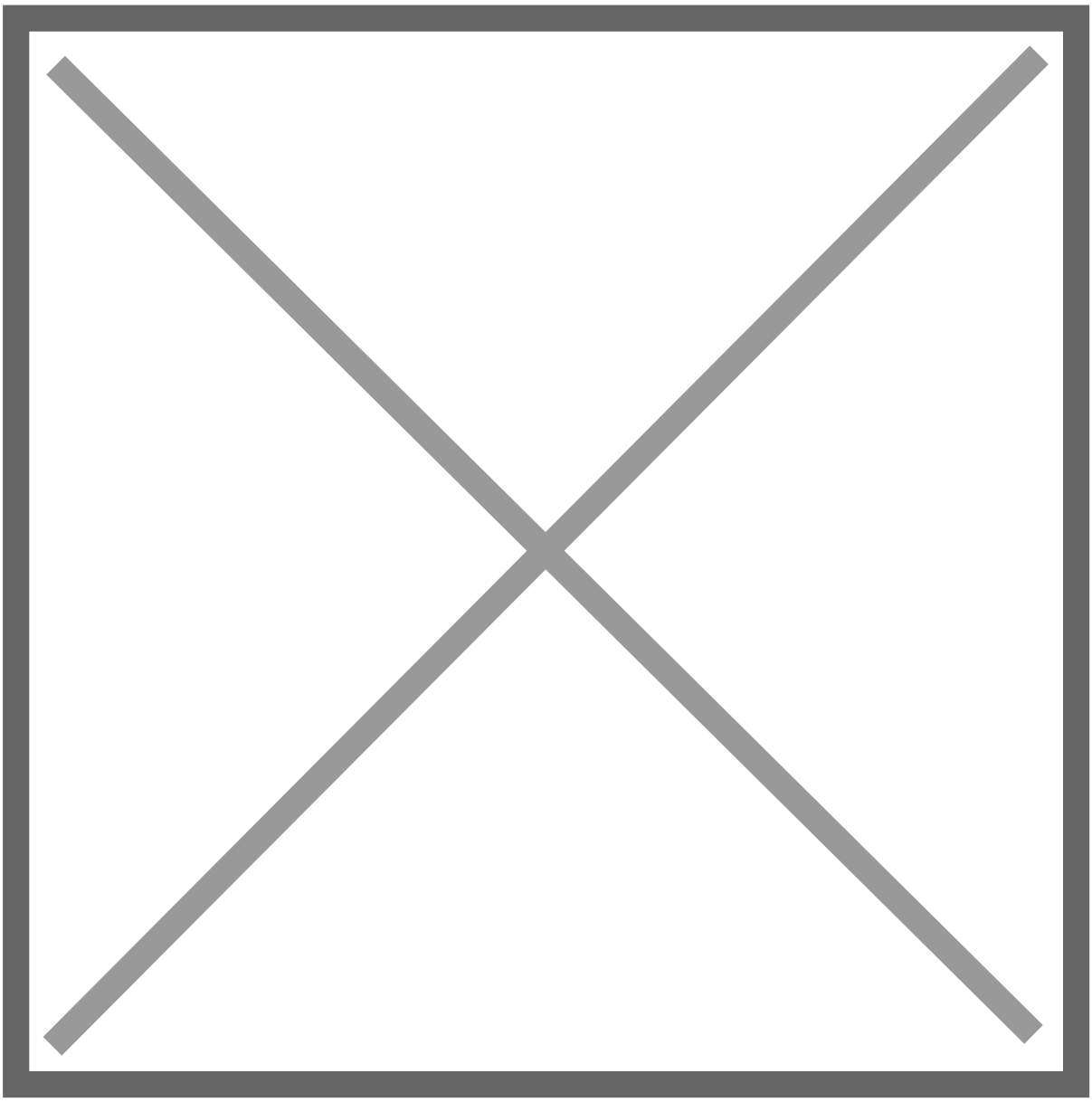
Sollte wget noch nicht installiert sein: sudo apt install wget

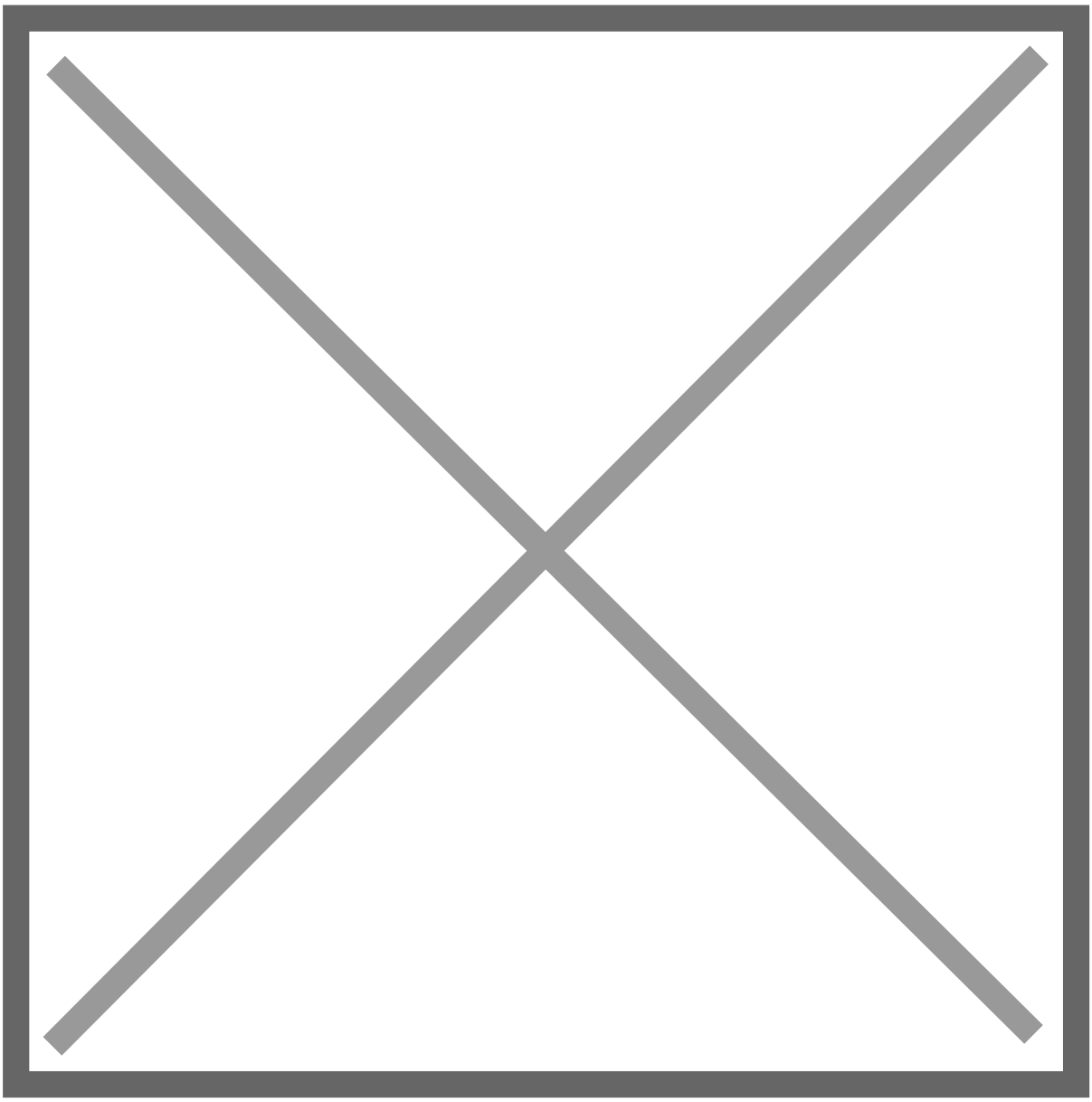
Der neue Raspiblitz ist nun fertig installiert. Diese soeben erstellte VM kannst du nach der erfolgreichen Installation herunterfahren.

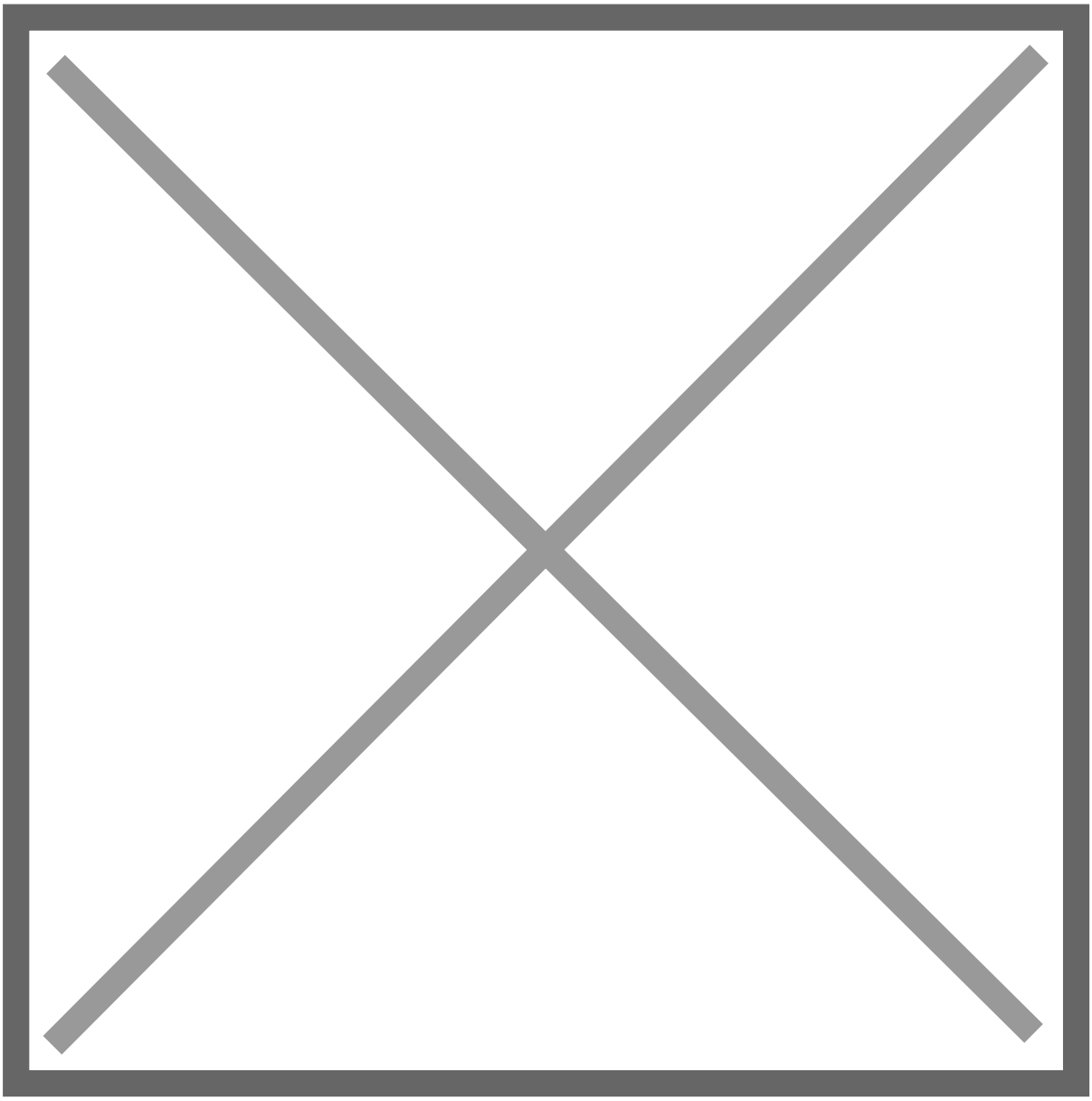
“Alter” Raspiblitz upgrade ready machen

Bevor du den alten Raspiblitz herunterfährst, wähle im Menü folgende Punkte aus:









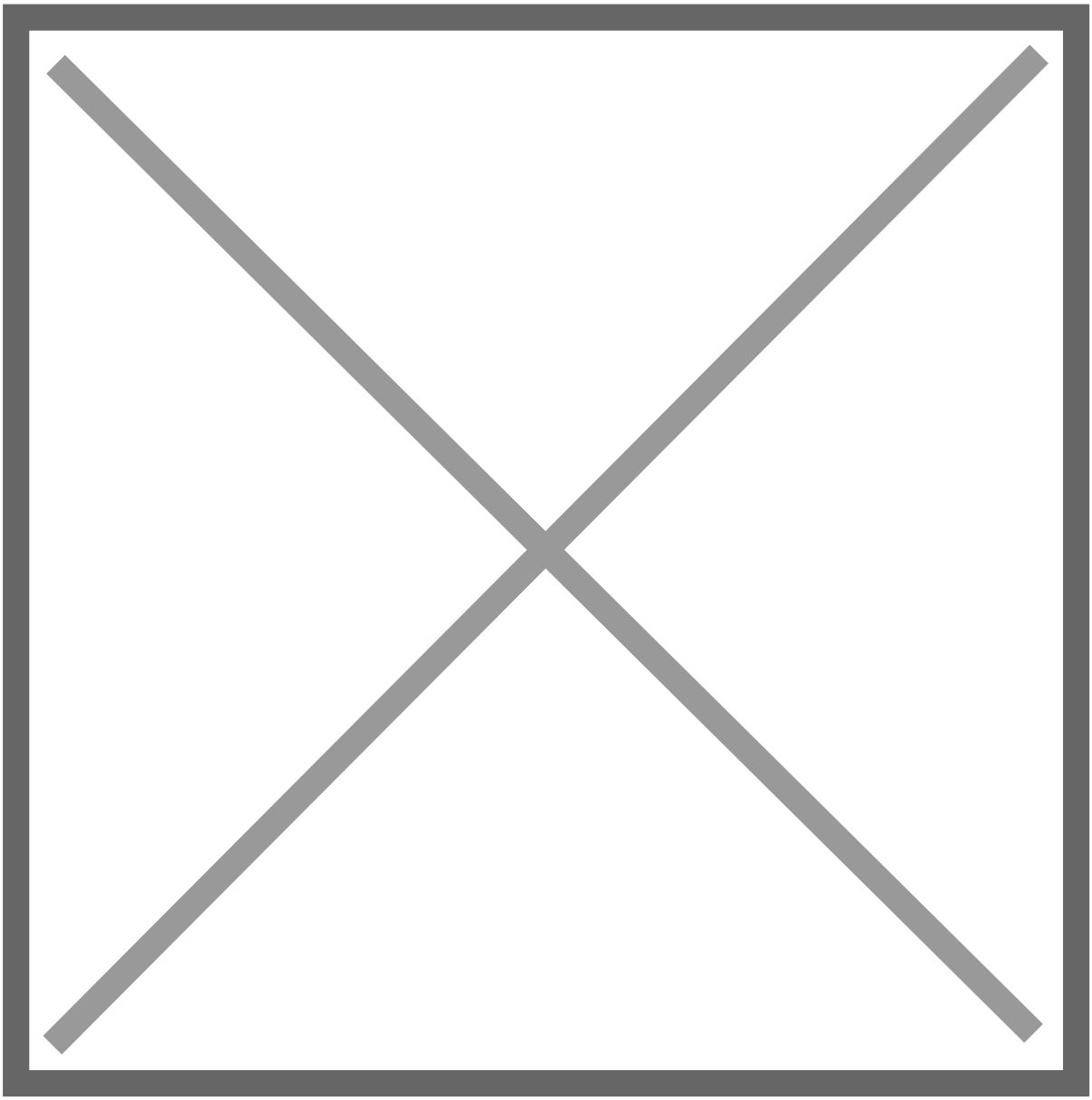
Danach wird der alte Raspiblitz ein Backup vom LND Ordner auf der externen Festplatte machen und sich danach herunterfahren.

Beide VM’s sind nun offline.

Festplatte in neue VM einhängen

~~Alte VM -> Hardware -> Harddisk (erkennbar an der Grösse) anklicken -> oben auf “Disk Action” klicken -> Reassign owner -> Die neue VM auswählen~~

Festplatte hier einbinden



Solltest du die “Start at boot” Option unter den Options gesetzt haben, ändere dies nun auf “no” bei der alten VM und auf “Yes” bei der neuen VM. Ansonsten würden beide VM’s bei einem Proxmox Neustart gleichzeitig starten.

Wenn du die gleiche IP Adresse bei der neuen VM trotz DHCP haben willst, empfehle ich dir die MAC Adresse unter Hardware -> Network Device von der alten VM zur neuen VM zu kopieren. Der DHCP Server erkennt dann die neue VM als die alte und vergibt ihr die gleiche IP. Solltest du mit einer DHCP Reservierung arbeiten, empfehle ich dir das umso mehr, da du so die Reservierung nicht bearbeiten musst.

Starte nun die neue VM, verbinde dich mit den default Anmeldeinformationen (user: admin, pw: raspiblitz) auf die neue VM. Du wirst als erstes aufgefordert, die Passwörter zu setzen. Du kannst da die gleichen die beim alten Raspiblitz verwenden.

Durchgereichte Festplatte

Wenn du für die Blockchain etc. keine virtuelle Festplatte hast, sondern eine physische Harddisk, die direkt an die VM durchgereicht wird, kannst du HIER in dieser Anleitung die Befehle ganz normal nochmals anwenden, als würdest du ein neuer Raspiblitz erstellen:

```
ls -n /dev/disk/by-id/
/sbin/qm set [VM-ID] -virtio2 /dev/disk/by-id/[DISK-ID]
```

Wichtige Punkte:

Achte darauf, dass beide VM’s nicht gleichzeitig laufen. Gerade wenn du die Mac Adresse von der alten zur neuen kopiert hast, kann dies zu Netzwerk Problemen führen. Sollte der Upgrade Prozess erfolgreich gewesen sein, kannst du die alte VM löschen.

Einloggen auf dem Raspiblitz per SSH

Benutzer: admin
Passwort: raspiblitz

Auf Proxmox USB durchschleifen für FHEM

> Proxmox: USB Passthrough an LXC Container

> Für diesen Vorgang ist ein Skript erstellt das im Normalfall 2 Minuten nach einem Neustart des Servers aktiviert werden sollte. Nach Möglichkeit demnächst mal überprüfen

Zusätzlich zum Proxmox und Proxmox: LXC Tutorial möchte ich heute gerne zeigen, wie man einen USB Stick zu einem Container durchreicht. Ich habe so meinen UZB1 Z-Wave USB Stick zu einer LXC mit Ubuntu und FHEM durchgereicht. Analog dazu kannst du jeden CUL oder andere USB-Geräte durchreichen.

Inhalt

Vorbereitungen in Proxmox

Als erster wählen wir uns über SSH in unseren Proxmox Server ein.

```
ssh root@192.168.178.XX
```

Nun stoppen wir den LXC Container, an den das USB-Gerät durchgereicht werden soll

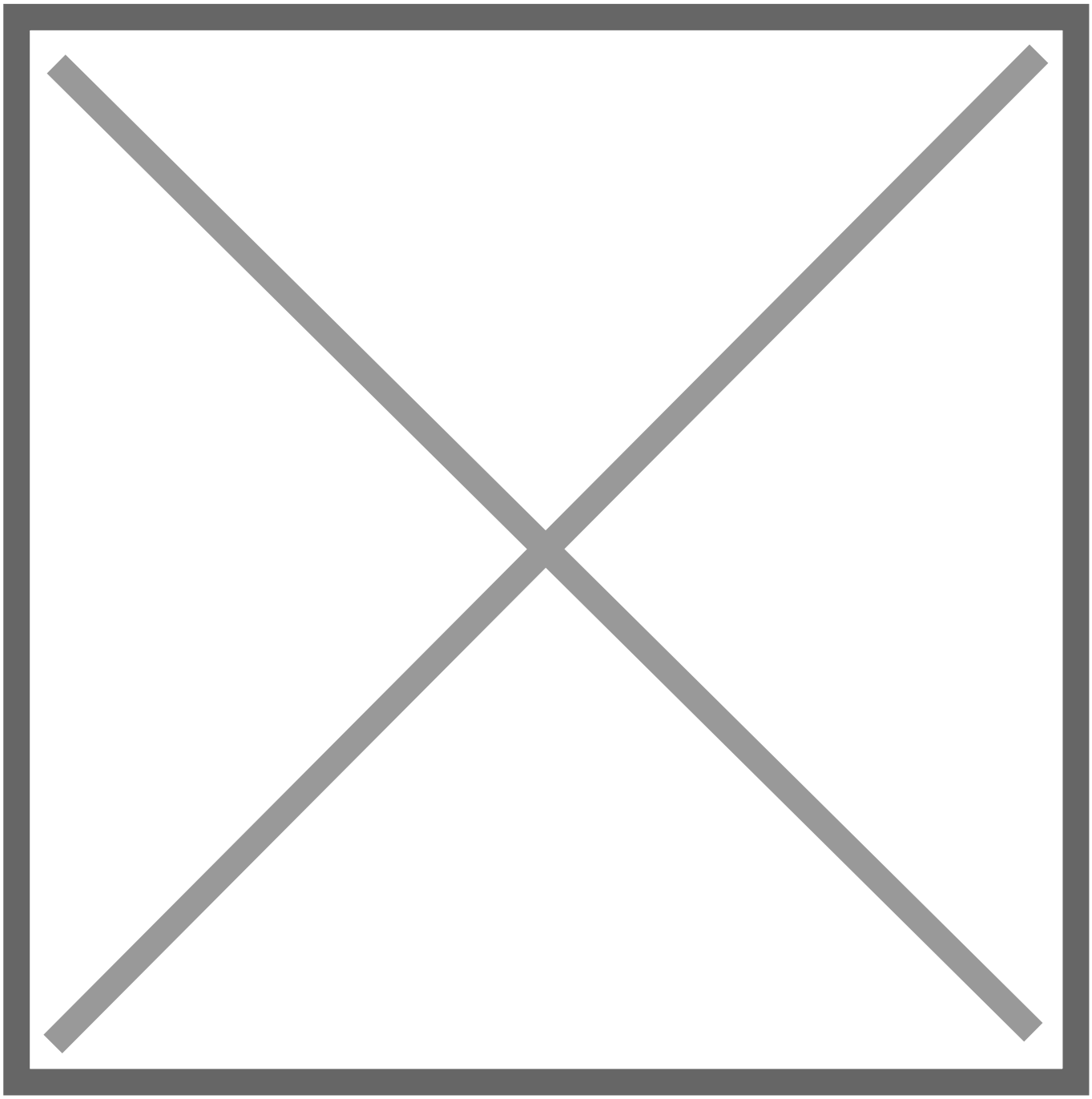
```
pct stop 109
```

Die Nummer enspricht deiner VM / LXC – ID in Proxmox. Diese findest du im Proxmox Webinterface heraus

Mit folgendem Befehl kannst du nun die verbundenen USB-Geräte auflisten. Es empfiehlt sich zuerst eine Auflistung ohne eingesteckten USB-Stick anzufertigen und dann nochmal, nachdem der Stick eingesteckt wurde

```
lsusb
```

Wie du auf dem folgendem Screenshot erkennen kannst, heißt mein UZB1 Stick „Sigma Designs, Inc. und hat die BUS Nummer 1 und Device Nummer 18. Du hast vermutlich eine andere Nummer -> notiere dir diese.



> Suche nach einem Device mit Future im Namen für Eltako

Mittels folgendem Befehl kannst du nun den DeviceTree anzeigen. Wichtig: Trage hier deine BUS und DEVICE Nummer ein

```
ls -l /dev/bus/usb/001/018
```

Es werden erneut zwei Zahlen ausgegeben, wobei nur die erste für uns relevant ist

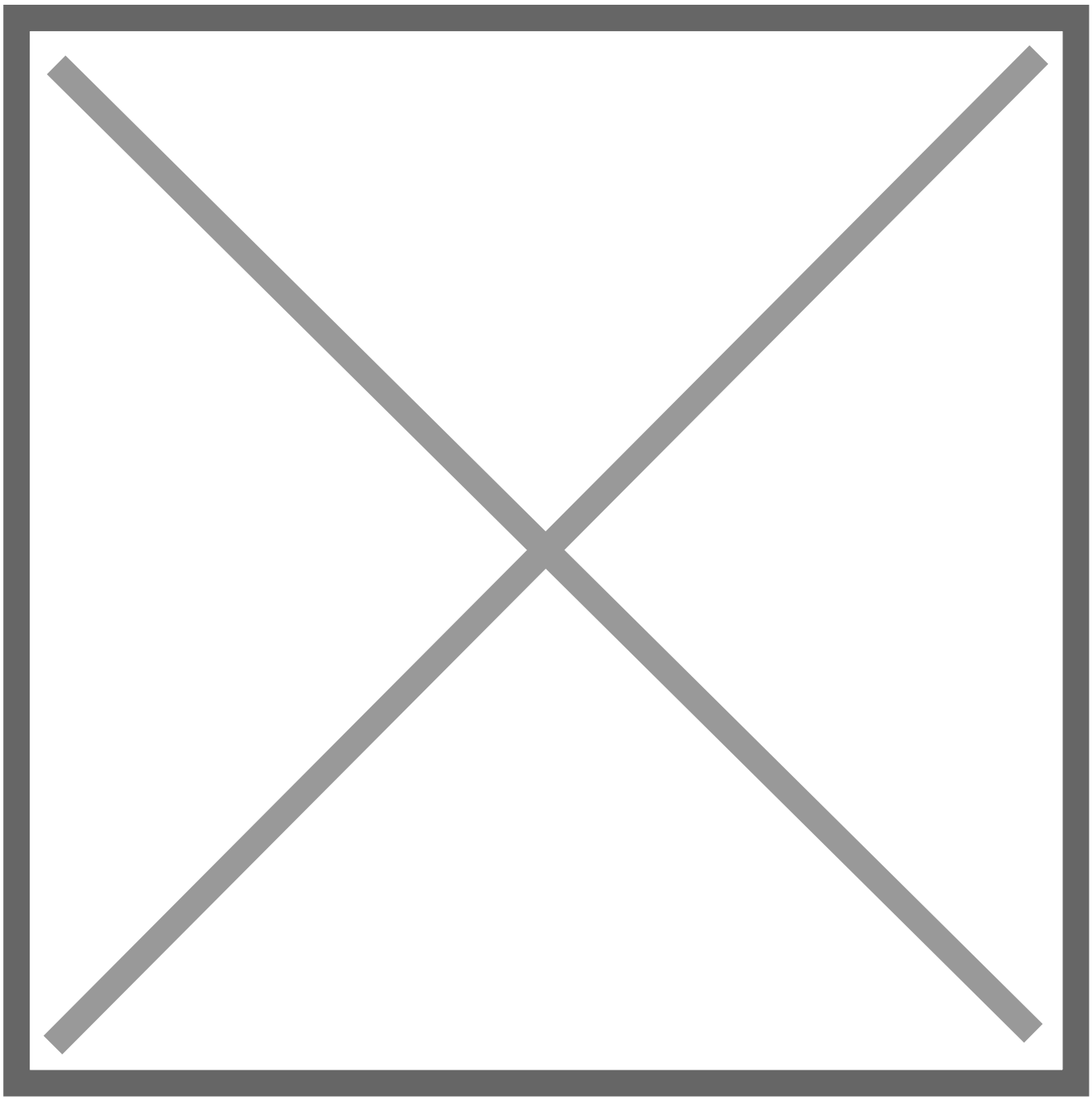
`crw-rw-r– 1 root root 189, 17 Jul 12 18:35 /dev/bus/usb/001/018`

Die Nummer 189 notierst du dir ebenfalls. Als nächstes müssen wir die Konfigurationsdatei für den LXC Container finden und editieren. Höchstwahrscheinlich findest du deine Konfigurationsdatei unter /etc/pve/nodes/pve/lxc – Schaue dazu am besten noch im Proxmox Webinterface welche ID dein Container hat.

Mit folgendem Befehl wechselst du in den Ordner mit den Konfigurationsdateien

```
cd /etc/pve/nodes/pve/lxc
```

mit „ls“ kannst du dir dann den Ordnerinhalt anzeigen lassen



```
nano 113.conf
```

Hier fügen wir folgende Zeilen am Ende der Datei ein

```
lxc.cgroup.devices.allow: c 189:* rwm
lxc.mount.entry: /dev/bus/usb/001/018 dev/bus/usb/001/018 none bind,optional,create=file
```

Trage hier entsprechend die BUS und Devicenummer, sowie die Nummer bei der Ausgabe von ls -l /dev/bus ..... ein

In meinem Fall mit dem UZB1 hat das aber noch nicht ausgereicht. Es wird ein Gerät mit dem Namen ttyACM0 angelegt. Dieses müssen wir ebenfalls durchreichen

Führe folgenden Befehl aus und notiere dir wieder die erste Nummer

```
ls -l /dev/ttyUSB0
```

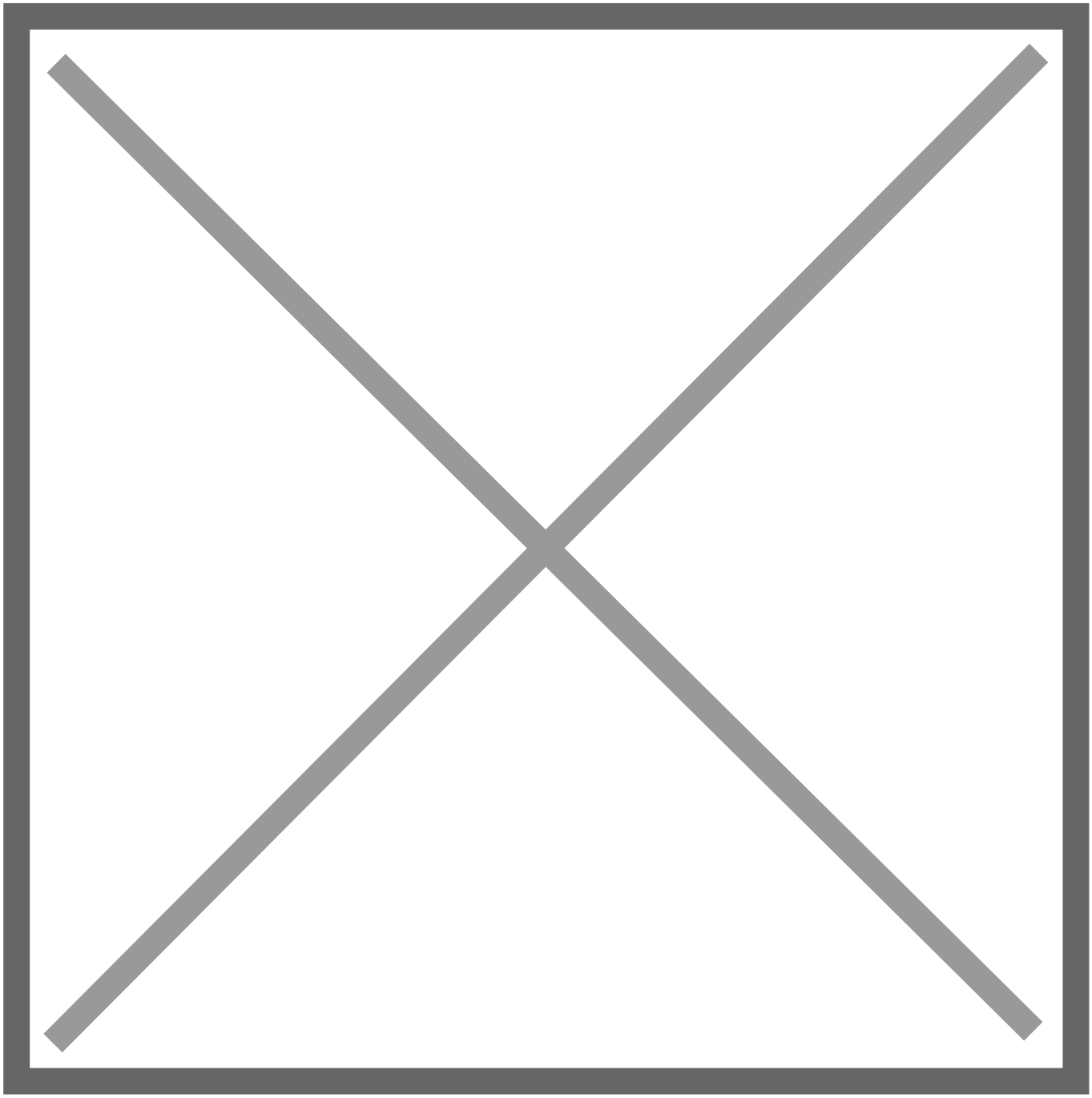
Dann bearbeiten wir nochmal die LXC Konfig-Datei:

```
sudo nano 109.conf
```

Und fügen am Ende nochmal zwei Zeilen hinzu

```
lxc.cgroup.devices.allow: c 166:* rwm
lxc.mount.entry: /dev/ttyACM0 dev/ttyACM0 none bind,optional,create=file
```

Am Ende sieht die Konfig-Datei in etwas so aus:



Dann statten wir das Gerät ttyACM0 noch mit den nötigen Rechten aus

```
chmod o+rw /dev/ttyUSB0
```

Zuletzt starten wir den LXC Container erneut und können im nächsten Punkt überprüfen, ob alles soweit geklappt hat

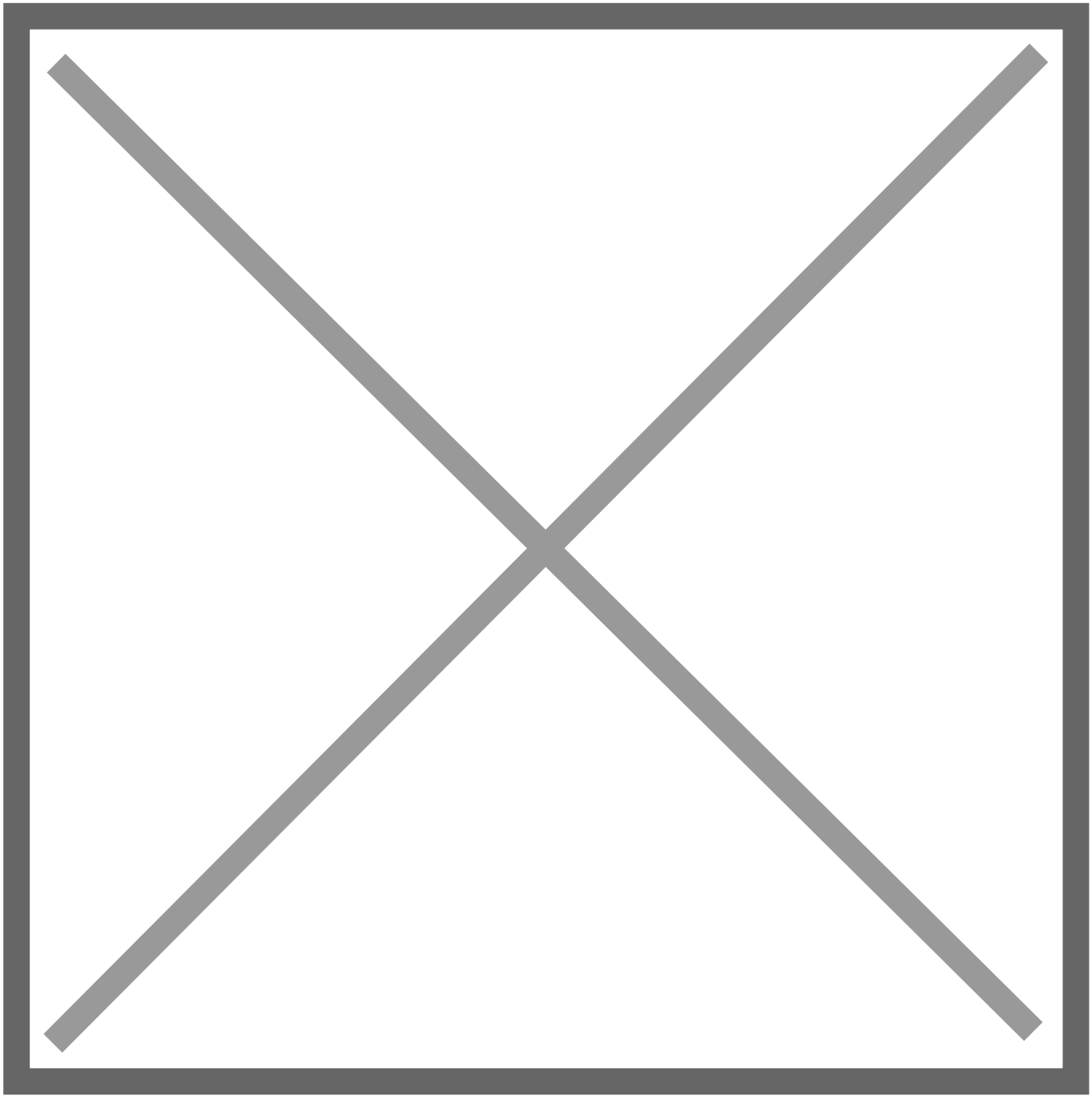
```
pct start 113
```

Überprüfung im Container

Verbinde dich mittels SSH mit deinem LXC

```
ssh NUTZERNAME@IP-ADRESSE
```

mit „lsusb“ kannst du dir auch hier die USB-Geräte anzeigen lassen. Ob das Gerät ttyACM0 angelegt wurde erfährst du unter den Devices:



Shell Screenshot

```
cd /dev
```

```
ls
```

Fazit

Zugegeben, ein wenig kompliziert ist es leider schon, allerdings funktioniert es wirklich einwandfrei. Wenn du dieses Tutorial auch bei anderen Geräte erfolgreich getestet hast würde ich mich über einen kurzen Kommentar freuen und könnte dies diesem Artikel hinzufügen.

Raspiblitz auf Proxmox installieren

Raspiblitz auf Proxmox installieren

Hier möchte ich euch aufzeigen, wie ihr einen neuen Raspiblitz auf einer Debian VM auf Proxmox installieren und in Betrieb nehmen könnt. Mein Raspiblitz lief sehr lange und stabil auf einem Raspberry Pi 4 mit 8GB RAM. Das würde er sehr wahrscheinlich auch noch länger so tun, jedoch wächst mein Lightning Node immer wie mehr und diverse Apps und Services bauen auf meinem Node auf. So wird das Thema der Verfügbarkeit und des Backups immer wie wichtiger. Deshalb habe ich entschieden, den Raspiblitz auf eine VM in Proxmox zu migrieren. So habe ich bezüglich Backup und Administration viel mehr Spielraum.

Dieser Guide hier wird euch helfen, ein komplett neuer Raspiblitz mit Proxmox aufzusetzen. Der Guide für die Migration folgt demnächst...

Was wird benötigt?

- Proxmox Installation auf einem Intel NUC, Laptop oder Server
- mind. 1TB SSD

Bei der SSD habt ihr mehrere Möglichkeiten: Entweder ihr baut die 1TB SSD im System fix ein und installiert dort drauf auch euer Proxmox Host oder (so wie ich es gemacht habe) ihr habt eine interne SSD (in meinem Fall 500GB M2 SSD) wo sich das Hostbetriebssystem drauf befindet. Die 1TB SSD habe ich anschliessend mittels SATA am Intel NUC bei mir angeschlossen. Diese wird ausschliesslich für das Speichern der Blockchain und Lightning Node bei mir verwendet.

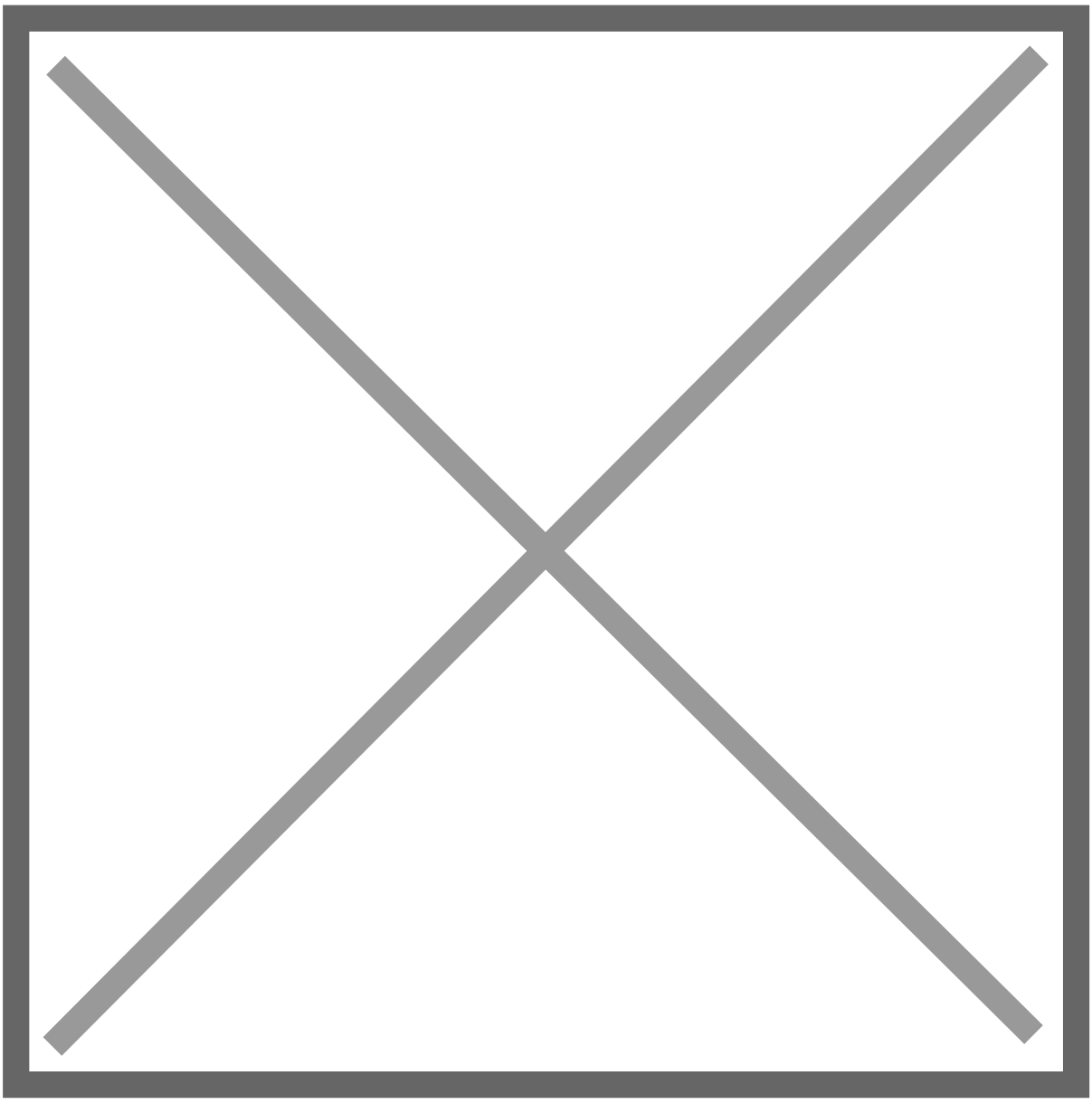
Debian VM erstellen

Wir installieren Raspiblitz auf einer frischen Debian Maschine. Deshalb müssen wir zuerst das ISO File von Debian herunterladen. Hier einfach die richtige Prozessor Architektur auswählen: (Für mich ist es amd64)

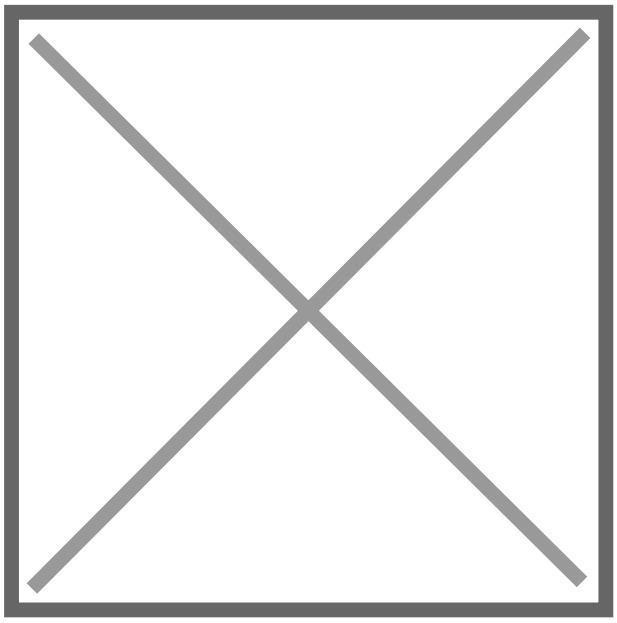
https://www.debian.org/distrib/

https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-12.5.0-amd64-netinst.iso

Danach kann diese ISO Datei unter Proxmox hochgeladen werden. Dazu auf den Local Storage klicken, ISO Images und auf Upload:

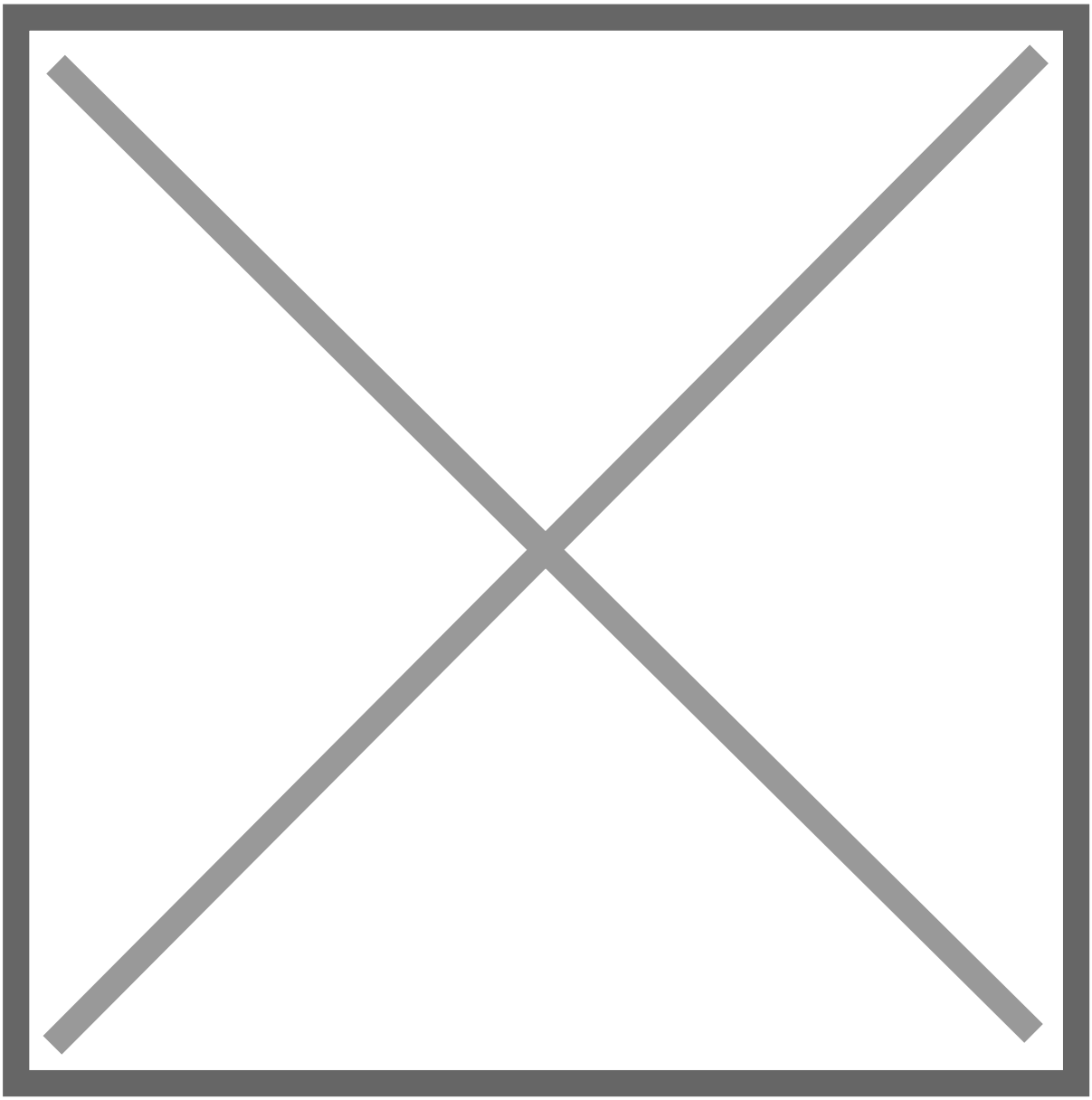


Nun kann man mittels “Create VM” oben rechts eine neue virtuelle Maschine erstellen. Da klicken wir nun drauf.

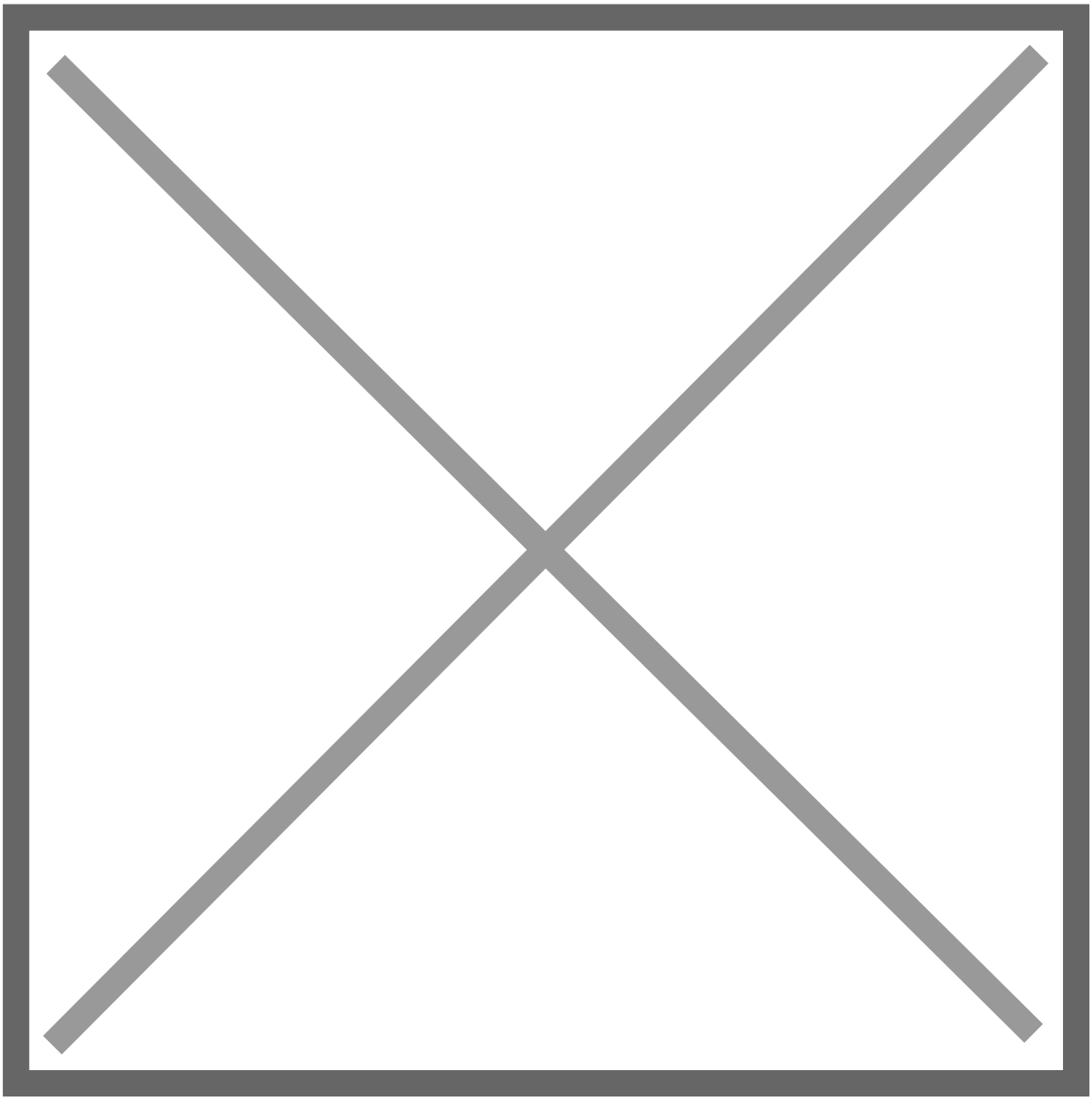


Es erscheint ein Einstellungsfenster, wo wir die Eigenschaften der VM nun angeben können. In meinem Beispiel sehen die Eigenschaften wie folgt aus:

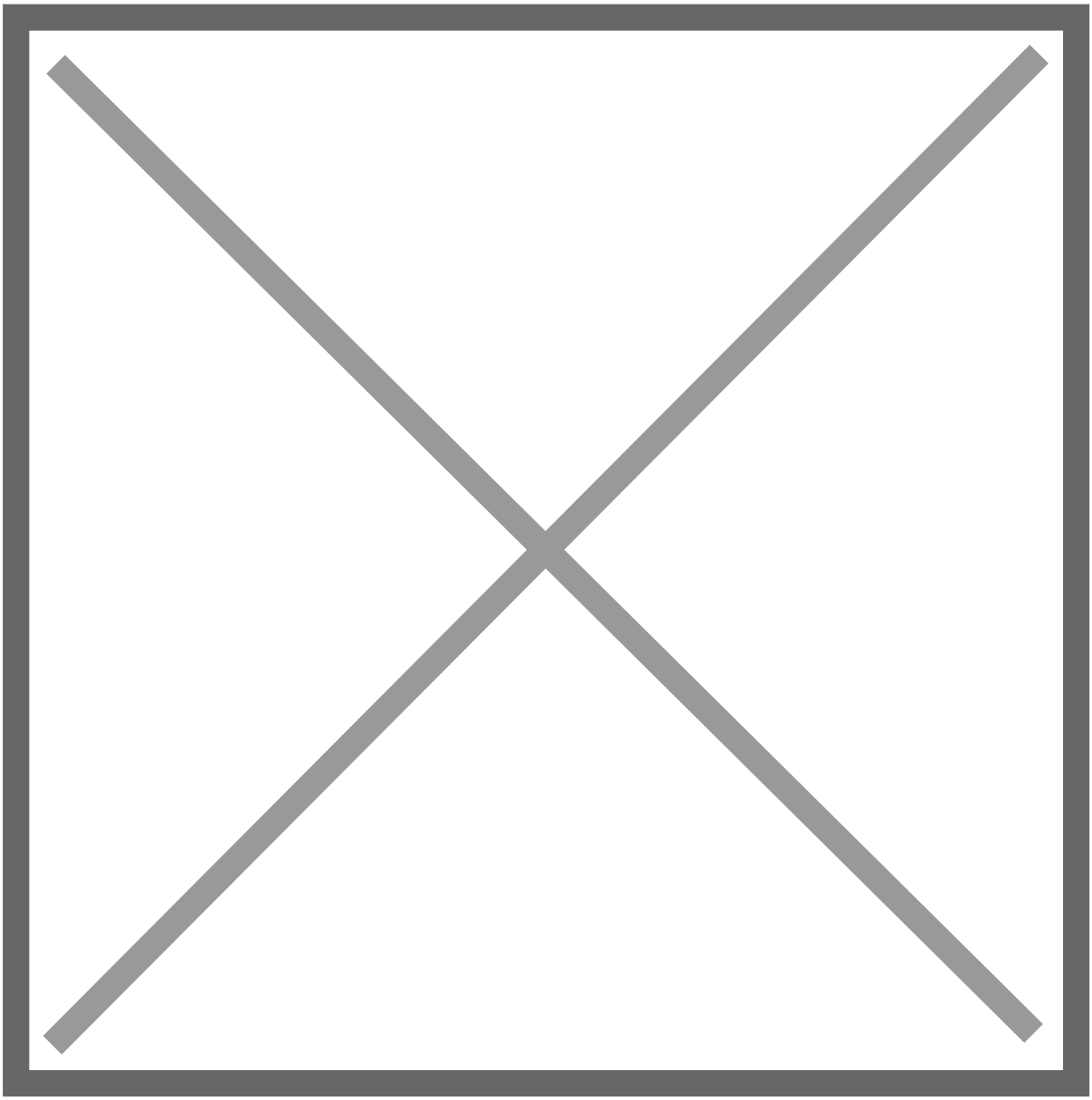
VM ID und Name kann selber ausgewählt werden.



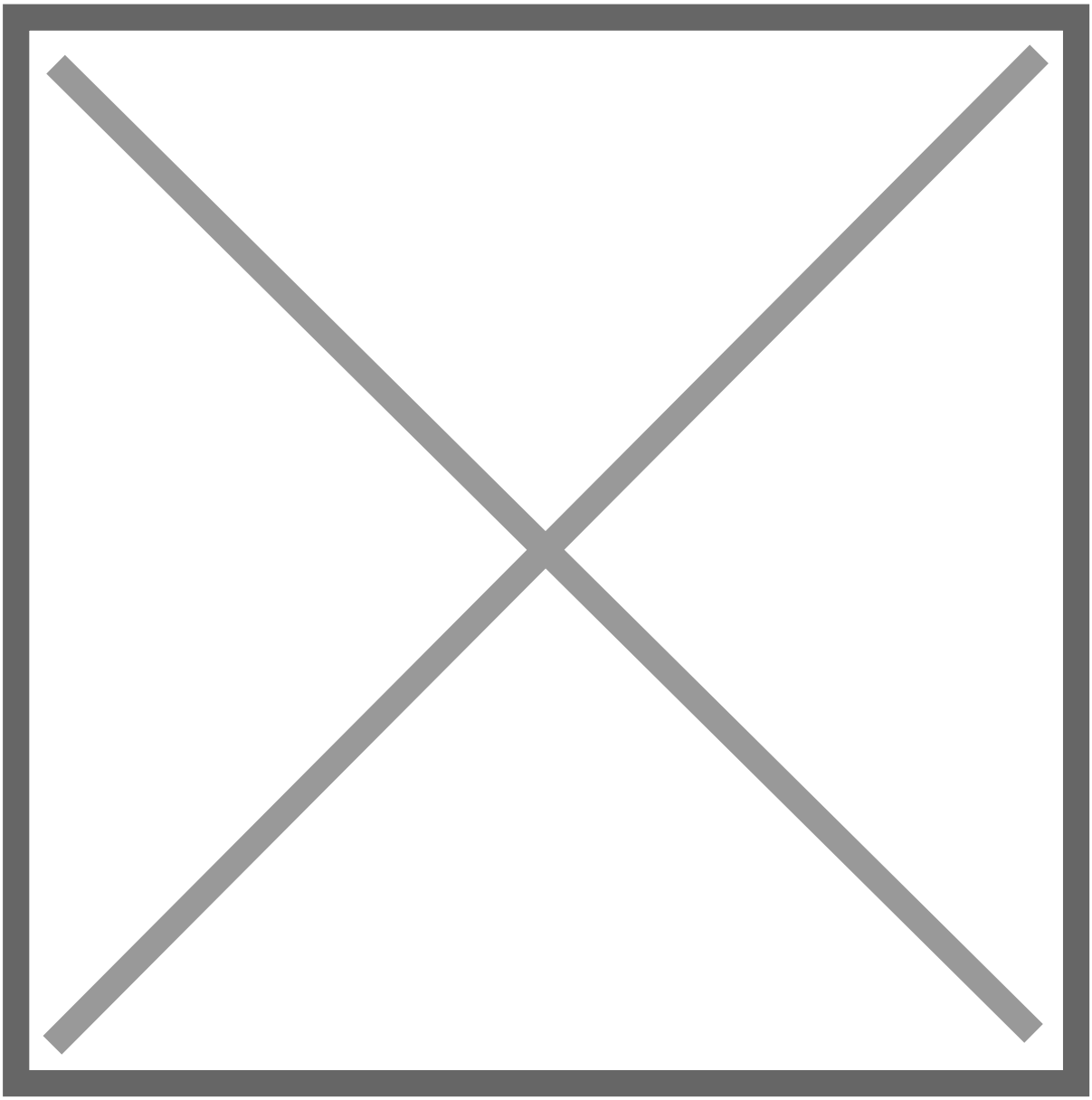
Unter dem Punkt “OS” wählen wir nun das vorher heruntergeladene ISO File aus:



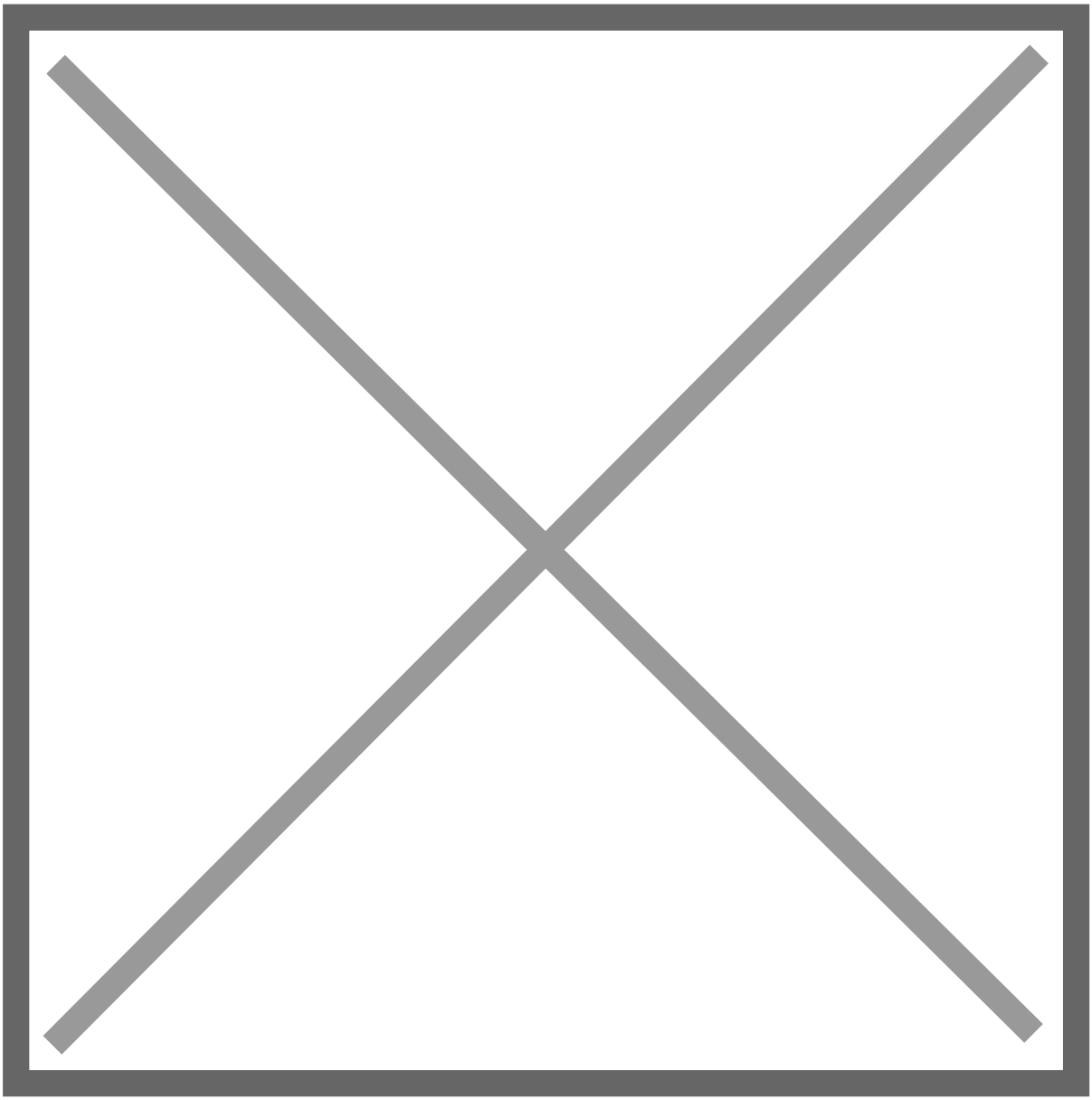
Unter “System” können wir die Default Einstellungen belassen. Ich habe hier noch den Hacken bei Qemu Agent gesetzt, damit die VM anschliessend mittels Qemu Agenten mit dem Proxmox Host kommunizieren und Daten übermitteln kann.



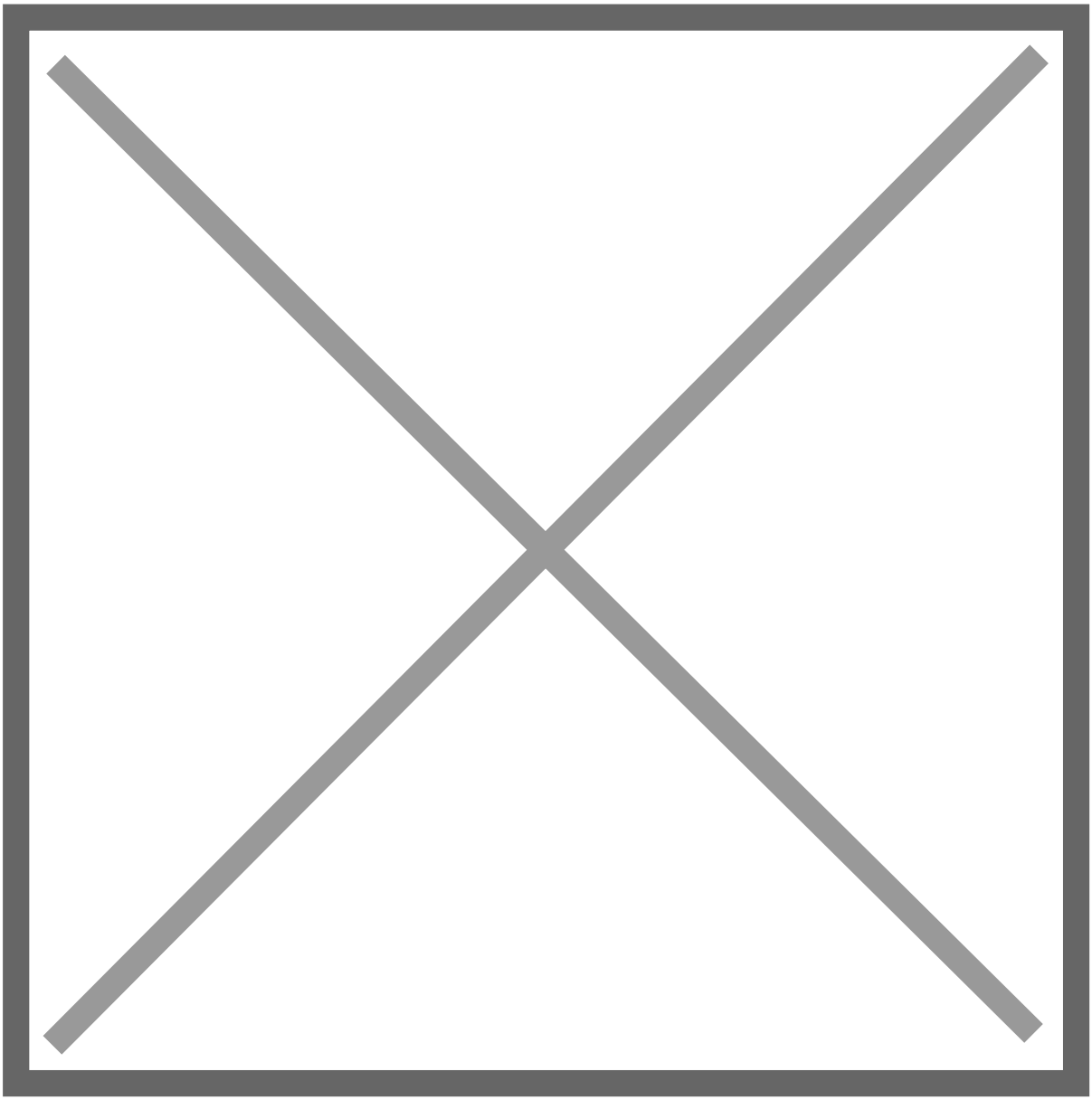
Unter “Disks” könnt ihr nun die gewünschte Grösse der VM angeben. Ich habe hier mal die gleiche Grösse (32GB) eingestellt, wie auch meine SD Karten vom Raspiblitz gross sind. Diese kann bei Bedarf zu jeder Zeit in der Zukunft vergrössert werden, falls man doch mehr Platz benötigt und die Host Maschine auch diesen Speicherplatz zur Verfügung hat. Ein weiterer Vorteil warum ich meinen Raspiblitz virtualisiert habe □□



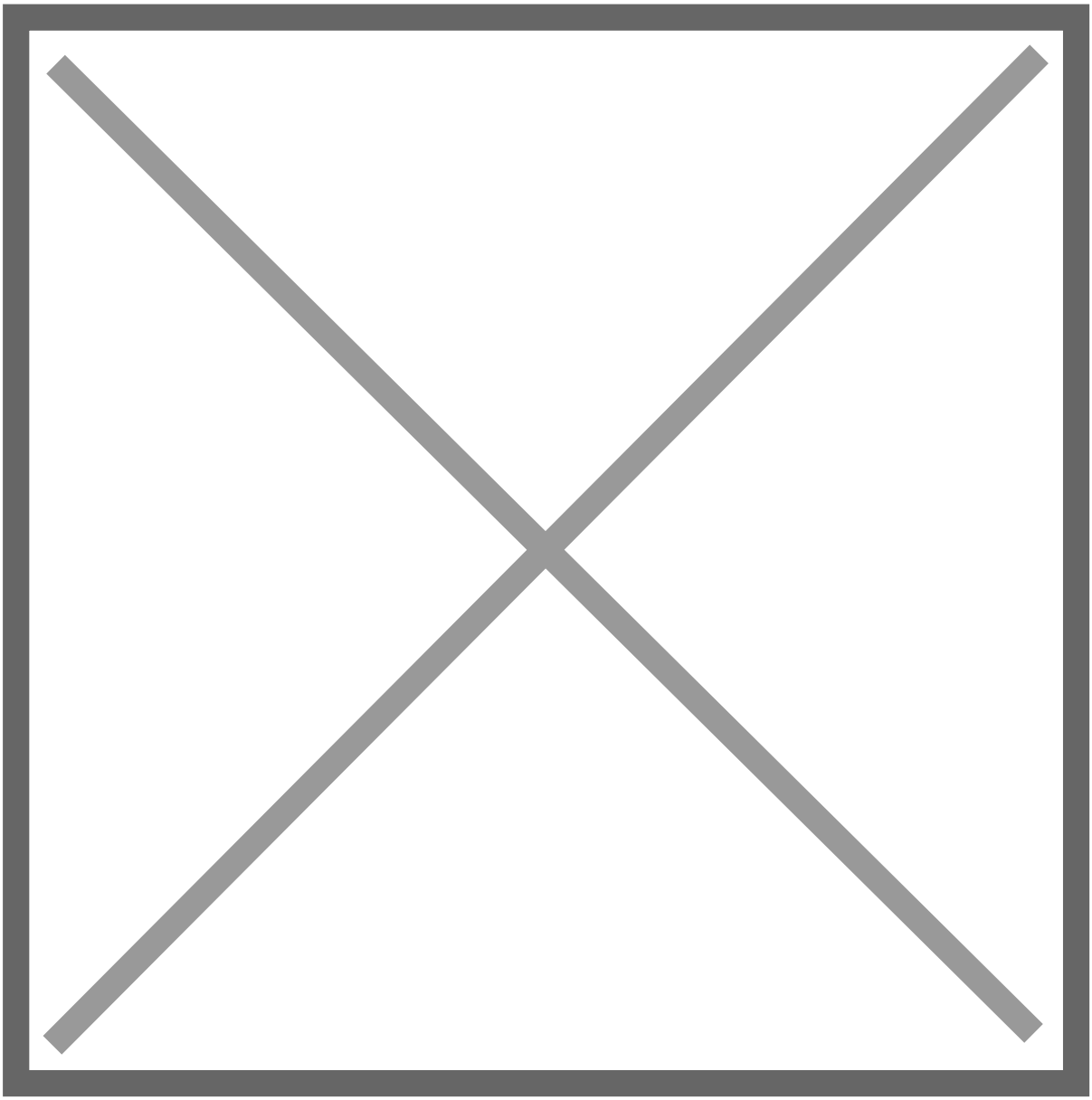
Unter “CPU” könnt ihr die gewünschte Core Anzahl angeben. Dies ist natürlich von eurem Hostbetriebssystem abhängig. Mein Intel NUC hat 4 Kerne, deshalb kann ich hier der VM auch gleich 4 zur Verfügung stellen.



Unter “Memory” müsst ihr die gewünschte RAM Anzahl in MB angeben. Auch dies ist von eurem Host abhängig. Wenn möglich, würde ich hier 8GB oder mehr eintragen. **Kleiner Hinweis: 1GB = 1024 MB. Somit sind 8GB = 8192 MB. (8 x 1024)**



Die VM braucht auch noch einen Netzwerkadapter. Dieser müsst ihr ebenfalls anhand eurer Proxmox Installation entsprechend auswählen. Standard wird vmbr0 sein. Auf meinem Proxmox habe ich diverse VLAN mit konfiguriert, wobei der vmbr3 mein BTC VLAN ist. Dies ist aber nur bei mir so.



Anschliessend könnt ihr auf “Finish” klicken und die VM wird erstellt. Diese taucht auch nun mit Namen auf der linken Seite auf und kann nun gestartet werden. (Rechtsklick -> Start)

Nun könnt ihr die Console (oben rechts) der VM öffnen und die Debian Installationsschritte normal durchführen. Ich habe die Schritte hier ein wenig abgekürzt:

- Install
- Sprache auswählen
- Location auswählen
- Keyboard Sprache auswählen
- Hostname setzen
- Domain setzen (oder einfach leer lassen)
- Root Passwort vergeben
- Neuer User erstellen (meiner heisst hier “pi”)
- Passwort für User pi vergeben

- Guided – use entire disk
- SCSI3 Harddisk auswählen
- All files in one partition
- Finish partitioning and write changes to disk
- “Write the changes to disks?” -> Yes
- Scan extra installation media? -> No
- Package manager -> Dein Land auswählen
- Package manager -> deb.debian.org
- http proxy -> leer lassen und continue
- Participate in the package usage survey? -> No
- Software selection: SSH server und standard utilities sollten hier ausreichen
- Install the GRUB boot loader to your primary drive? -> Yes
- /dev/sda auswählen

Die VM ist nun fertig installiert und beginnt zu starten. In der Zwischenzeit kann man nun die ISO Datei entfernen. (VM -> Hardware -> CD/DVD Drive -> Do not use any media -> OK)

Speicher hinzufügen

Um Raspiblitz nun auf dieser neuen VM zu installieren, müssen wir 2 Sachen machen: Festplatte für die Blockchain Daten anschliessen und der VM durch reichen und das SD Card Builder Script von Raspiblitz installieren. Wir beginnen zuerst mit der Festplatte, wobei es hier 2 Varianten gibt:

Variante 1: Externe Festplatte

Schliesst nun die Festplatte mittels SATA oder USB beim Hostsystem an. In meinem Beispiel hier benutze ich eine USB Festplatte, welche ich mittels USB 3.1 am Intel NUC angeschlossen habe. Am besten fährt ihr die Raspiblitz VM nun herunter.

Nun müsst ihr euch bei der Konsole auf dem Proxmox Hostsystem anmelden und folgende Schritte ausführen:

Die Befehle aus dem Video hier noch einmal zum kopieren:

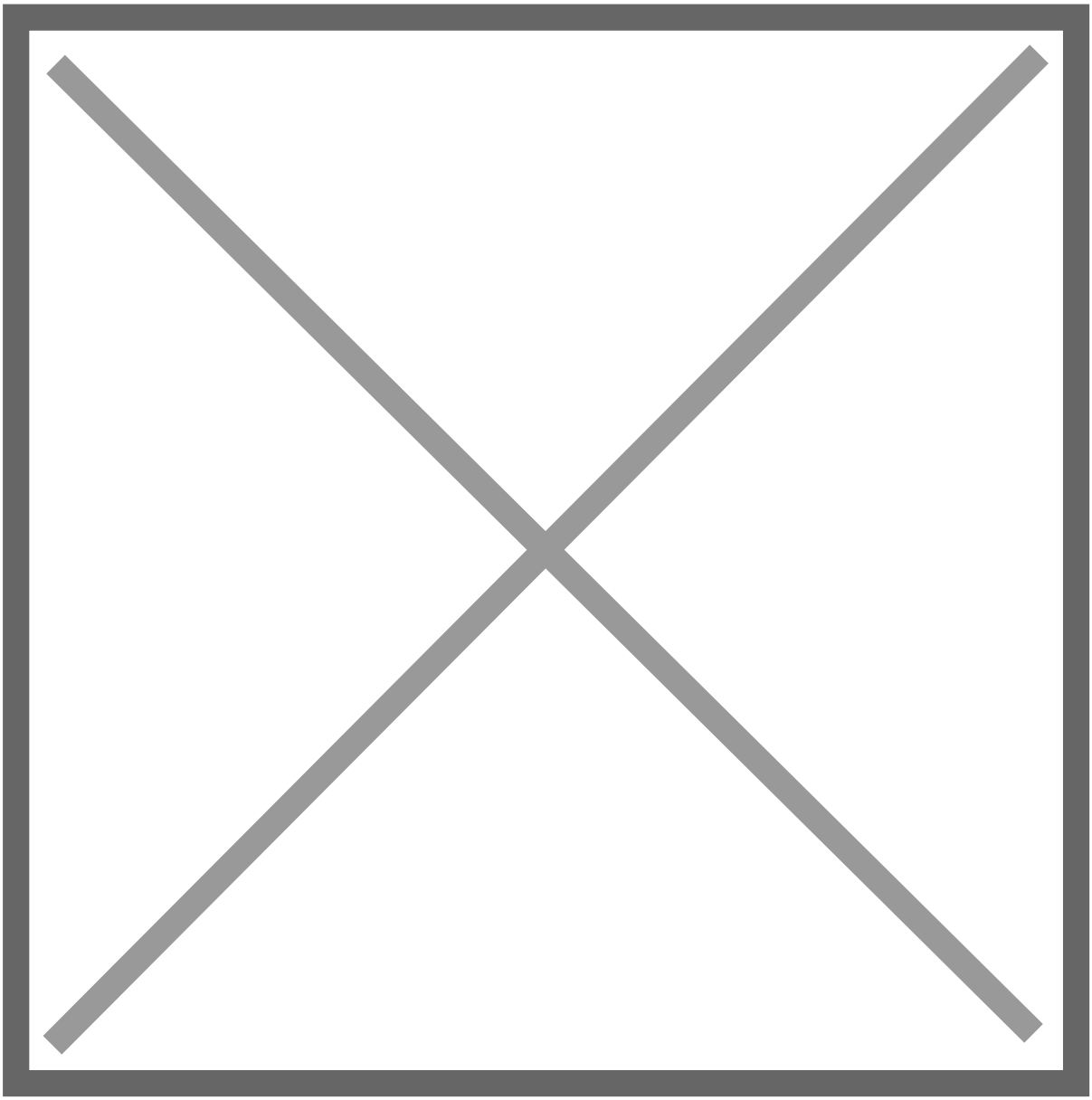
```
ls -n /dev/disk/by-id/
/sbin/qm set [VM-ID] -virtio2 /dev/disk/by-id/[DISK-ID]
```

Wichtig ist hier, dass die Festplatte mittels ID durch gereicht wird. Sollte sich mal etwas in Zukunft an der dev sda Reihenfolge ändern, wird immer noch die richtige Festplatte mit der VM verbunden.

Variante 2: Interner Speicherplatz verwenden

Falls ihr genügend Speicherplatz auf dem Hostbetriebssystem habt, müsst ihr nicht zwingend eine externe Festplatte dazu verwenden. Hier könnt ihr einfach der VM eine zweite Festplatte unter “Hardware -> Add -> Hard Disk” hinzufügen. Ich würde hier mind. 1TB als Speichergrösse empfehlen.

Egal ob Variante 1 oder 2 ausgeführt wurde, die VM sollte nun in der Hardware Übersicht 2 Festplatten verbunden haben: Eine kleinere (z.B. 32GB) wo das Betriebssystem von Raspiblitz drauf installiert und laufen wird und eine grössere (z.B. 1TB oder mehr) wo die ganzen Daten der Blockchain später abgespeichert werden.



Raspiblitz installieren

Nun sind wir ready, um Raspiblitz mittels Script zu installieren. Dafür starten wir die Raspiblitz VM und loggen uns als root Benutzer in der Konsole ein. Zuerst einmal alles updaten:

```
apt update
apt upgrade -y
apt install sudo
```

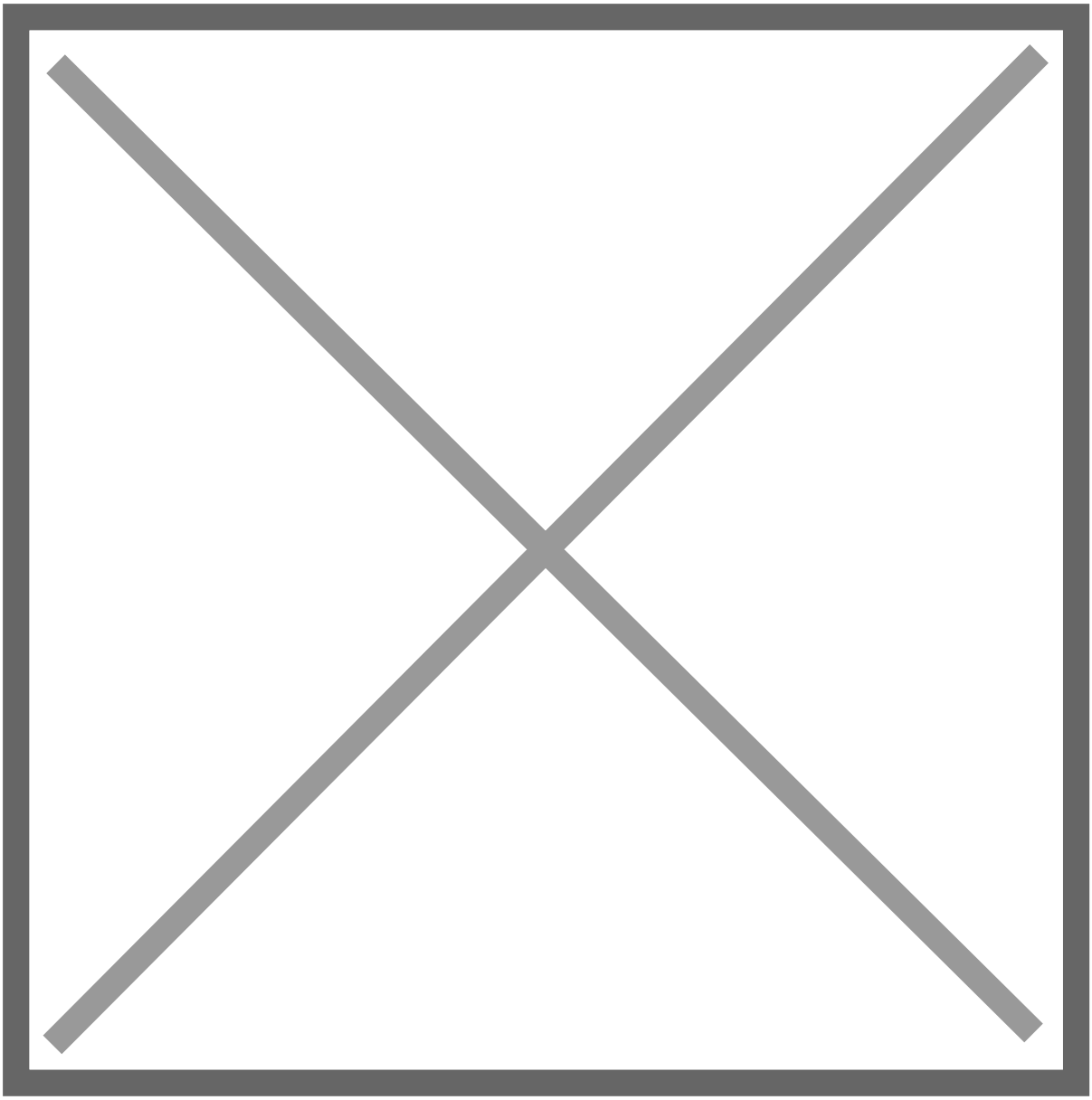
Nun müssen wir das Build SDCard Script von Rootzoll herunterladen. Die Version kann beliebig angepasst werden. Die aktuellste Version ist die 1.11.

```
wget https://raw.githubusercontent.com/raspiblitz/raspiblitz/v1.11/build_sdcard.sh
```

Und ausführen:

```
sudo bash build_sdcard.sh -f false -d headless -t false -w off
```

Das Script zeigt dir nun Informationen von deinem System an. Wenn das alles stimmt, mit “yes” starten.



Nun dauert die Installation einige Minuten. Hier nicht abbrechen oder die VM herunterfahren, sondern einfach installieren lassen. Wenn alles abgeschlossen ist, wird ein reboot benötigt:

```
sudo shutdown -r now
```

Nun könnt ihr im Browser die IP Adresse von eurer VM aufrufen und die ganz normalen Installationsschritte von Raspiblitz durchführen.

Festplattenübersicht

Externe USB Festplatte	2 TB	sdc	Für Nextcloud Daten	/mnt/Daten
Local SSD	1 TB	sda		/mnt/local-2
Local NVME	1 TB	nvme0n1	VM Daten	local-lvm 850 GB local 100 GB
	2 TB	sdb	USB-Festplatte bitcoin	/mnt/Daten-Bitcoin

LXC Container Fix Docker Problem

```
# Config Datei via Proxmox Host aufrufen
nano /etc/pve/lxc/<id>.conf

#hinzufügen
lxc.apparmor.profile: unconfined
lxc.mount.entry: /dev/null sys/module/apparmor/parameters/enabled none bind 0 0

# LXC neu starten
pct reboot <id>
```

Debian Template erstellen

Vorbereitung der VM

Konfigurieren Sie die Debian-VM vollständig: Installieren Sie benötigte Pakete (z. B. SSH-Server, Cloud-Init für bessere Flexibilität: `apt install cloud-init`), passen Sie Netzwerk, Benutzer und Systemeinstellungen an. Löschen Sie dann maschinen-spezifische Daten wie `/etc/machine-id` (`truncate -s 0 /etc/machine-id && systemd-machine-id-setup`), SSH-Host-Keys (`rm /etc/ssh/ssh_host_*`) und führen Sie `cloud-init clean` aus, falls Cloud-Init verwendet wird. Fahren Sie die VM herunter.

Template erstellen

Wählen Sie die VM im Proxmox-Webinterface aus, klicken Sie auf **More** > **Convert to template**. Die VM wird in eine nicht-startbare Vorlage umgewandelt (Icon ändert sich). Optional: Fügen Sie vorher ein Cloud-Init-CD-Laufwerk hinzu (`qm set <VMID> --ide2 local-lvm:cloudinit`), um bei Klonen IP, SSH-Keys usw. automatisch zuzuweisen.

Neue VM bereitstellen

Klicken Sie rechts auf die Template > **Clone**. Wählen Sie **Full Clone** (vollständige Kopie, unabhängig) oder **Linked Clone** (speichersparend, aber abhängig vom Template). Starten Sie die neue VM und finalisieren Sie einzigartige Einstellungen wie Hostname. Mit Cloud-Init können Sie vor dem Start IP (`qm set <neueVMID> --ipconfig0 ip=192.168.1.100/24,gw=192.168.1.1`) und SSH-Keys setzen.

VM-Vorbereitung schrittweise

Ja, die Vorbereitung einer Debian-VM für ein Proxmox-Template erfolgt systematisch innerhalb der VM, um maschinen-spezifische Daten zu entfernen und Wiederverwendbarkeit zu gewährleisten.

1. **Debian-VM starten und konfigurieren**: Installieren Sie notwendige Pakete wie `apt update && apt install openssh-server cloud-init qemu-guest-agent`. Passen Sie Einstellungen an (Hostname generisch halten, z. B. `hostnamectl set-hostname template`, Netzwerk auf DHCP oder statisch minimal).
2. **Maschinen-ID leeren**: Führen Sie `truncate -s 0 /etc/machine-id && systemd-machine-id-setup` aus, damit jede neue VM eine einzigartige ID erhält.
3. **SSH-Keys regenerieren**: Entfernen Sie `rm /etc/ssh/ssh_host_*` und starten Sie SSH neu (`systemctl restart ssh`), damit Klonen neue Keys erzeugt. Deaktivieren Sie optional Root-Login (`sed -i 's/PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config`).

4. **Cloud-Init bereinigen (empfohlen)**: Installieren Sie `cloud-init` falls nicht vorhanden, dann `cloud-init clean --logs --seed`. Erstellen Sie ein Cloud-Init-Laufwerk in Proxmox: `qm set <VMID> --ide2 local-lvm:cloudinit`.
5. **Swap und Logs aufräumen**: Deaktivieren Sie Swap falls nicht benötigt (`swapoff -a`), leeren Sie Logs (`logrotate -f /etc/logrotate.conf`) und führen Sie `apt autoremove && apt clean` aus.
6. **VM herunterfahren**: `shutdown -h now`. Im Proxmox-UI dann **More > Convert to template** auswählen.

Diese Schritte sorgen für saubere Klone mit automatischer Anpassung via Cloud-Init. Testen Sie mit einem Klon, ob SSH und Netzwerk funktionieren.

SSH muss wieder hergestellt werden

Korrekte Reihenfolge für Templates

Die Keys **müssen nach dem Klonen** in der **neuen VM** neu generiert werden, nicht im Template:

1. **Im Template-Vorbereitung**: Keys entfernen (`sudo rm /etc/ssh/ssh_host_*`), **aber SSH NICHT neu starten** und VM herunterfahren.
2. **Nach Klonen/Start der neuen VM**:

   text
   `sudo dpkg-reconfigure openssh-server`
   oder

Backup Proxmox

pve-config-backup.sh

Sichert die gesamte Konfiguration eines Proxmox-Hosts in ein komprimiertes Archiv (~10MB). Kein vollständiges Disk-Image — nur die relevanten Konfigurationsdateien.

Inhalt des Backups

Pfad	Was wird gesichert
`/etc/pve`	Alle VM/CT-Konfigurationen, Storage, Benutzer, Cluster, Netzwerk
`/etc/network/interfaces`	Bridge- und Netzwerkkonfiguration
`/etc/network/interfaces.d`	Zusätzliche Netzwerk-Includes
`/etc/hosts`	Hostname-Auflösung
`/etc/hostname`	Hostname des Systems
`/etc/fstab`	Einhängepunkte (z.B. NFS, lokale Disks)
`/etc/resolv.conf`	DNS-Konfiguration
`/etc/default/grub`	GRUB-Parameter inkl. IOMMU (`intel_iommu=on`)
`/etc/default/grub.d`	Zusätzliche GRUB-Konfiguration
`/etc/modprobe.d`	Kernel-Modul-Konfiguration (VFIO, Passthrough)
`/etc/modules`	Autostart-Kernel-Module
`/etc/udev/rules.d`	Udev-Regeln (USB-Persistenz, Disk-Mapping)
`/etc/systemd/system`	Eigene Systemd-Dienste und Timer
`/etc/cron.d`	System-Cronjobs
`/etc/cron.daily`	Tägliche Cronjobs
`/etc/cron.weekly`	Wöchentliche Cronjobs

Pfad	Was wird gesichert
`/etc/cron.monthly`	Monatliche Cronjobs
`/var/spool/cron/crontabs/root`	Root-Crontab
`/root`	SSH-Keys, eigene Skripte, `.bashrc`, `.profile`

Was ebenfalls abgedeckt ist

- **USB-Passthrough**: Konfiguration liegt in `/etc/pve/qemu-server/<vmid>.conf`
- **Disk-Passthrough**: Ebenfalls in den VM-Configs unter `/etc/pve/qemu-server/`
- **IOMMU/VFIO**: `/etc/default/grub` + `/etc/modprobe.d/`
- **Udev-Persistenz-Regeln** für USB-Geräte: `/etc/udev/rules.d/`

Installation

```
# Skript auf Proxmox kopieren
scp pve-config-backup.sh root@<proxmox-ip>:/root/
chmod +x /root/pve-config-backup.sh
```

Manueller Aufruf

```
# Standard (Ziel: /mnt/pve/ds-woehr-neu/panzerbackup)
/root/pve-config-backup.sh

# Abweichendes Zielverzeichnis
BACKUP_DIR=/mnt/anderes/ziel /root/pve-config-backup.sh

# Mehr Backups behalten (Standard: 7)
KEEP=14 /root/pve-config-backup.sh
```

Automatischer Betrieb (Systemd Timer)

Service und Timer installieren

```
# Dateien kopieren
scp pve-config-backup.service root@<proxmox-ip>:/etc/systemd/system/
scp pve-config-backup.timer root@<proxmox-ip>:/etc/systemd/system/

# Aktivieren
systemctl daemon-reload
systemctl enable --now pve-config-backup.timer
```

Läuft täglich um **03:00 Uhr** mit bis zu 10 Minuten Zufallsversatz.

Status prüfen

```
# Nächster geplanter Lauf
systemctl list-timers pve-config-backup.timer

# Letzter Lauf und Ergebnis
systemctl status pve-config-backup.service

# Logs der letzten Läufe
journalctl -u pve-config-backup.service -n 50
```

Timer deaktivieren

```
systemctl disable --now pve-config-backup.timer
```

Backup-Dateien

Backups werden im Zielverzeichnis mit folgendem Namensschema gespeichert:

```
pve-config_<hostname>_<datum>_<uhrzeit>.tar.gz
```

Beispiel:

```
/mnt/pve/ds-woehr-neu/panzerbackup/
├── pve-config_proxmox_2026-03-07_03-00-12.tar.gz
├── pve-config_proxmox_2026-03-06_03-00-08.tar.gz
└── pve-config_proxmox_2026-03-05_03-00-15.tar.gz
```

Es werden standardmäßig die letzten **7 Backups** behalten, ältere werden automatisch gelöscht.

Neuestes Backup anzeigen:

```
ls -lt /mnt/pve/ds-woehr-neu/panzerbackup/pve-config_*.tar.gz | head -1
```

Wiederherstellen

Vorbereitung

Backup-Datei identifizieren:

```
ls -lt /mnt/pve/ds-woehr-neu/panzerbackup/pve-config_*.tar.gz
```

Inhalt des Archivs prüfen (ohne etwas zu ändern):

```
tar tzf /mnt/pve/ds-woehr-neu/panzerbackup/pve-config_proxmox_DATUM.tar.gz | less
```

Komplette Wiederherstellung nach Neuinstallation

1. **Proxmox frisch installieren** (gleicher Hostname empfohlen)
2. **Synology einbinden:**

```
mkdir -p /mnt/pve/ds-woehr-neu
mount -t cifs //172.16.1.4/pxxData /mnt/pve/ds-woehr-neu -o username=<user>,password=<pass>
```

3. **Backup einspielen:**

```
tar xzf /mnt/pve/ds-woehr-neu/panzerbackup/pve-config_proxmox_DATUM.tar.gz -C /
```

4. **GRUB aktualisieren** (wichtig wenn IOMMU-Einstellungen vorhanden):

```
update-grub
```

5. **Kernel-Module neu laden:**

```
update-initramfs -u
```

6. **Neustart:**

```
reboot
```

Nach dem Neustart sind alle VM/CT-Konfigurationen, Netzwerkeinstellungen, Passthrough-Konfigurationen und eigene Dienste wiederhergestellt.

Einzelne Dateien wiederherstellen

Nur Netzwerkkonfiguration:

```
tar xzf pve-config_proxmox_DATUM.tar.gz -C / etc/network/interfaces
```

Nur VM-Konfiguration (z.B. VM 100):

```
tar xzf pve-config_proxmox_DATUM.tar.gz -C / etc/pve/qemu-server/100.conf
```

Nur GRUB-Konfiguration:

```
tar xzf pve-config_proxmox_DATUM.tar.gz -C / etc/default/grub
update-grub
```

Konfiguration per Umgebungsvariable

Variable	Standard	Beschreibung
`BACKUP_DIR`	`/mnt/pve/ds-woehr-neu/panzerbackup`	Zielverzeichnis
`KEEP`	`7`	Anzahl der aufzubewahrenden Backups

Anforderungen

- Bash
- `tar`, `gzip` (auf jedem Debian/Proxmox vorinstalliert)
- Schreibzugriff auf `BACKUP_DIR`
- Root-Rechte (wegen `/etc/pve` und `/root`)