STEUERUNGSRECHNER:

```
echo "Host r1
  HostName 192.168.10.11
  User root
Host u1
  HostName 192.168.10.11
  User user
Host dhcp1
  HostName 192.168.10.31
  User user
Host r2
  HostName 192.168.10.12
  User root
Host u2
  HostName 192.168.10.12
  User user
Host dhcp2
  HostName 192.168.10.32
  User user
Host r3
  HostName 192.168.10.13
  User root
Host u3
  HostName 192.168.10.13
  User user
Host dhcp3
  HostName 192.168.10.33
  User user" > /home/tchangalo/.ssh/config
```

```
echo "alias ipbc='ip -br -c a'
alias r1='ssh r1'
alias u1='ssh u1'
alias dhcp1='ssh dhcp1'
alias r2='ssh r2'
alias u2='ssh u2'
alias dhcp2='ssh dhcp2'
alias r3='ssh r3'
alias u3='ssh u3'
alias dhcp3='ssh dhcp3'" >> /home/tchangalo/.bashrc
```

Aktivieren mit: `. ~/.bashrc`

PVE INSTALLIEREN

Updaten

SSH-Key des Steuerungsrechners unter `/home/root/.ssh/authorized_keys` abspeichern.

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub root@rX
```

```
cd /root/.ssh
ssh-keygen -t ed25519    [ OHNE PASSPHRASE! ]

apt install -y sudo mc bpytop htop termshark lnav python3 python3.11 \
python3-pip python3.11-venv python3-venv jq
```

`ubuntu-live-server.iso` und `pfSense.iso` hochladen.

`vmbr1001` MGMT erstellen. Die IP von `vmbr1001` ist `10.20.30.254/24`

Später werden folgende weitere IP's im MGMT-Netz vergeben:
```
dhcp1 10.20.30.251
dhcp2 10.20.30.252
dhcp3 10.20.30.253
```

SKRIPTE KOPIEREN

Für X die Knoten-Nr. einsetzen.

Steuerungsrechner:

```
scp useradd.sh root@rX:/root
scp pfsenseX.sh root@rX:/root
scp pfsX_postinstall.sh root@rX:/root
scp dhcpX.sh root@rX:/root
scp dhcpX_postinstall.sh root@rX:/root
```

USER UND .VENV ANLEGEN

pve-root: `./useradd.sh`

Steuerungsrechner: `ssh-copy-id -i ~/.ssh/id_ed25519.pub user@uX`

```
su - user
cd /home/user/.ssh
ssh-keygen -t ed25519    [ OHNE PASSPHRASE! ]

sudo visudo  ==>
Defaults timestamp_timeout=40
```

oder (nicht empfohlen!): `%sudo   ALL=(ALL:ALL) NOPASSWD: ALL`

PFSENSE INSTALLIEREN

2 CPUs, 1536 MB RAM , 8 GB Festplatte

pve-root: `./pfsenseX.sh`

VM 1000 starten und installieren. Statt Auto (ZFS) nehmen wir Auto (UFS) und dann GPT statt MBR.
Swap rauslöschen, um SSD nicht kaputt zu machen!

Am Ende nicht 'Reboot' wählen, sondern 'Shell' und dann `poweroff` eingeben.

pve-root: `./pfsX_postinstall.sh`

Falls möglich vor dem ersten Start über die MAC Adresse (z.B. unter Network Device -> MAC Address) im Router eine feste IP für die pfSense definieren: Z.B. 192.168.10.21 für pfSense1, 192.168.10.22 für pfSense2 und 192.168.10.23 für pfSense3.

Default Login: user: admin passwd: pfsense

Unter System / Advanced / Networking 'Kea DHCP' auswählen.

Unter den WAN Firewall Regeln die Anti-Lockout Regel replizieren mit Protocol: any, Destination: WAN address.

pfSense runterfahren.

Unter PVE die vlan-sensiblen Anschlüsse vmbr1, vmbr2 und/oder vmbr3 – ohne irgendwelche IPs – anlegen.

Der pfSense einen neuen Network Device auf der `vmbr1` mit VLAN-Tag 1011 hinzufügen. Dabei `VirtIO (paravirtualised)` auswählen und den Haken von Firewall wegnehmen. Das ist das LAN1. Nach dem Hochfahren der pfSense dieses Device mit der IP `172.11.0.1/24` konfigurieren. Die DHCP-Server Range muss nicht besonders groß sein, z.B. `172.11.0.10` bis `172.11.0.20` .
Vor dem Hochfahren ggf. (je nach dem, ob man nur einen oder alle drei Provider auf dem PVE aufsetzen möchte) noch zusätzlich `vmbr2` und `vmbr3` anlegen und dabei ebenfalls `VirtIO (paravirtualised)` auswählen und den Haken von Firewall wegnehmen.
`vmbr2` wird in der pfSense zu LAN2 und bekommt die IP `172.12.0.1/24` und den VLAN-Tag 2011. `vmbr3` wird in der pfSense zu LAN3 und bekommt die IP `172.13.0.1/24` und den VLAN-Tag 3011.
Bei LAN2 und LAN3 auf EINEM Node (!) die 'Default allow LAN2(3) to any rule' anlegen: Protocol: any, Source: LAN2(3) subnets, Destination: any

UBUNTU DHCP-SERVER AUFSETZEN

1536 MB RAM, 4 Cores (host) und 8 GB Festplatte

Sofern man nur auf einem einzigen PVE-Node arbeitet, reicht es, nur dhcp1 aufzusetzen, der dann die IP’s für die Router von ISP2 und ISP3 mitvergibt. (In diesem Fall braucht man natürlich auch nur die pfSense1 mit vmbr1, vmbr2 und vmbr3.)

pve-root: `./dhcpX.sh`

Vor dem Starten über die MAC im Router feste IP `192.168.10.31[2,3]` für den DHCP-Server definieren bzw. falls das nicht geht, nach der Installation.
Vor dem Starten außerdem MGMT-Interface `net1` anlegen und dann während der Installation manuell die IP `10.20.30.251[2,3]/24` für `node1[2,3]` setzen.

Nach dem Installieren: Kein `reboot`, kein `shutdown`, sondern `stop` und dann:
`./dhcpX_postinstall.sh`

```
Steuerungsrechner: ssh-copy-id -i ~/.ssh/id_ed25519.pub user@dhcpX

sudo swapoff -a

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

sudo apt install mc termshark kea lnav -y

sudo mc  ==> Delete /swap.img

echo "alias ipbc='ip -br -c a'" >> /home/user/.bashrc
```

Aktivieren mit: `. ~/.bashrc`

```
sudo apt install qemu-guest-agent -y
sudo systemctl start qemu-guest-agent
sudo systemctl status qemu-guest-agent
```

Falls wenig RAM vorhanden: Ausschalten und RAM auf 1024 MB runtersetzen.

KEA-DHCP SERVER AUFSETZEN:

Steuerungsrechner: `scp kea-dhcp4.conf user@dhcpX:/home/user`

dhcp-user:

```
sudo mv /etc/kea/kea-dhcp4.conf /etc/kea/kea-dhcp4.conf.bak
sudo mv /home/user/kea-dhcp4.conf /etc/kea/
sudo chown root:root /etc/kea/kea-dhcp4.conf
```

Bei Node 2 und 3 muss die kea-dhcp4.conf angepasst werden:

```
sudo nano /etc/kea/kea-dhcp4.conf
```

"interfaces": ["ens19/10.20.30.251 bzw. 2 bzw. 3"]

```
sudo systemctl restart kea-dhcp4-server
sudo systemctl status kea-dhcp4-server

[journalctl -u kea-dhcp4-server]
```

VYOS CLOUD INIT IMAGE ERSTELLEN

https://docs.vyos.io/en/latest/automation/cloud-init.html

Das Vyos Cloud Init Image kann nicht unter PVE erstellt werden, sondern nur auf einer VM oder einem sonstigen Linux-Rechner!

```
apt update && apt full-upgrade
apt install curl git ansible
git clone https://github.com/vyos/vyos-vm-images.git
```

Versionsbezeichnung anpassen:
Statt der 'latest' die zweitneueste Version von `https://vyos.net/get/nightly-builds/` runterladen und unter dem Namen `vyos.iso` nach `/tmp` des erstellenden Rechners verschieben.

VM-root:
```
chown root:root /tmp/vyos.iso

cd vyos-vm-images

sed -i '/download-iso/p' qemu.yml

ansible-playbook qemu.yml -e disk_size=10 -e iso_local=/tmp/vyos.iso -e \
grub_console=serial -e vyos_version=1.5.0 -e cloud_init=true -e \
cloud_init_ds=NoCloud
```

Beim letzten Befehl nötigenfalls die Versionsnummer anpassen!

UNTER DOCKER

```
sudo wget https://raw.githubusercontent.com/vyos/vyos-vm-images/current/Dockerfile

sudo docker build --tag vyos-vm-images:latest -f ./Dockerfile .

sudo docker run --rm -it --privileged -v $(pwd):/vm-build -v $(pwd)/images:/images -w /vm-build \
vyos-vm-images:latest bash

sudo git clone https://github.com/vyos/vyos-vm-images.git && cd vyos-vm-images
```

Jetzt als root im Container bleiben!
Versionsbezeichnung anpassen:
Statt der 'latest' die zweitneueste Version auf Steuerngsrechner von `https://vyos.net/get/nightly-builds/` runterladen und unter dem Namen `vyos.iso` nach `/tmp` verschieben. Nötigenfalls Versions-Nr. anpassen!):

```
wget https://github.com/vyos/vyos-rolling-nightly-builds/releases/download/1.5-rolling- \
202406250020/vyos-1.5-rolling-202406250020-amd64.iso

mv vyos.iso /tmp/vyos.iso

ansible-playbook qemu.yml -e disk_size=10 -e iso_local=/tmp/vyos.iso -e \
grub_console=serial -e vyos_version=1.5.0 -e cloud_init=true -e \
cloud_init_ds=NoCloud
```

Das qcow2-Image sollte jetzt in /tmp liegen. Jetzt das erstellte Image in ein Volume kopieren, das auf den Host gemappt ist (in Portainer nachsehen).

SEED.ISO ERSTELLEN

Dazu sind drei Dateien nötig: network-config, user-data und meta-data. Diese Dateien sind unter `/streams/seed/` zu finden. Die SSH-Credentials in `user-data` anpassen!

Die seed.iso wird mittels folgendem Befehl erstellt:

```
mkisofs -joliet -rock -volid "cidata" -output seed.iso meta-data user-data network-config
```

Die `seed.iso` in den regulären ISO-Images Ordner verschieben, d.h. nach `var/lib/pve/local-btrfs/template/iso .`

INTERNET CREATOR STARTEN

Nach dem Clonen dieses Repos den Ordner `streams` aus dem Ordner `internet_creator_<latest>` herausnehmen und in den Pfad `/home/user/` des PVE-Hosts ablegen und dann von da aus arbeiten.
Der Ordner `ansible` darf nicht world-writable sein, also z.B. 770.

Nun die neueste Version von Vyos rolling runterladen und unter `/home/user/ansible/vyos-files/` ablegen (dazu den Ordner vyos-files anlegen). Diese Datei ab und zu aktualisieren, damit das Update auf dem neuesten Stand bleibt.

Die Datei `generate_secret_key.py` in VSCode ausführen (Python Extension installieren, wenn nicht schon vorhanden) und den Secret Key in Zeile 6 von `app.py` einfügen, z.B:

```
6  app.secret_key = '\x0c\xf6\xb0\x00\x80%\xf0\xaf\x13\xec\xe0\xc6R\x90\
xeeh\xb1\xfe\x95\x93\x92\x7f\xaa\xa3'
```

Und dann eingeben:

```
source .venv/bin/activate
cd streams
python inc.py
```

Jetzt kann der Network Creator im Browser aufgerufen werden unter: **`<ip-pve>:5000`**

SKRIPTE AUSFÜHREN

Hier einige Beispiele, wie die Skripte zur Erstellung der Router per CLI gestartet werden können, um eine Vorstellung davon zu bekommen, was jeweils in die einzelnen Felder der GUI einzutragen ist:

VM's erstellen und konfigurieren:

`./provider.sh 1 1 1 1 "-l p1r1v" 1` // r1v von p1 erstellen mit Refresh.

`./provider.sh 1 1 1 2 "-l p1r1v,p1r2v" 1` // r1v und r2v von p1 erstellen mit Refresh.

`./provider.sh 1 1 1 4 "-l p1r[1-4]v" 1` // r1v bis r4v von p1 erstellen mit Refresh.

`./provider.sh 1 1 1 9 "" 0` // r1v bis r9v von p1 erstellen ohne Refresh.

`./provider.sh 2 2 1 9 "" 0` // r1v bis r9v von p2 erstellen ohne Refresh.

`./provider.sh 2 2 1 2 "-l p2r[1-2]v" 0` // r1v und r2v von p2 erstellen ohne Refresh.

`./provider.sh 2 2 1 6 "-l p2r[1-6]v" 0` // r1v bis r6v von p2 erstellen ohne Refresh.