Homelab

Release 1

Jan Jansen

May 11, 2025

CONTENTS:

CHAPTER

ONE

INTRO

This doc does not belong here, but did not want to create multiple github repo's.

This is a personal account of venturing into the Proxmox virutualised world:

- running containers
- running local LLM
- no GPU
- cheap

I try to document as much as I can, to avoid solving the same problem twice.

1.1 homelab

I bought hp dl380p g8 (generation8) with 16 cores and 32 threads and 256 GIGA bytes of RAM!! for the price of a raspberry pi.

I plan it on using during winter, so its heat is not lost. During rendering or running a LLM it generates 400 Watt. (or consumes for 400 watt expensive electricity)

It could use a GPU, but I hate to spend more money, and it would need a special riser to give way to PCIe x16 double slot.

results sofar :

- LLM runs at 5 tokens / second (llavafile)
- blender takes half an hour for rendering (CYCLES) a single picture
- using it as a ramdisk give 1,5G/s readspead!

1.2 software

I choose proxmox as a virtualisation platform, it runs linux containers (LXC), which are kind of cool since they are created quickly, launched quickly. Some problems arise since there are still shared resources with the host… hence the use of the included templates

CHAPTER

TWO

UPGRADING HOMELAB HP DL 380P

2.1 latest BIOS

- download latest firmware (BIOS) for linux in RPM format
- firmware-system-p70-2019.05.24-1.1
- on ubuntu (unpack with Ark)
- look for CPQP7013.6B8 (4MB in size)
- use ilo to upload firmware (update)

2.2 M2 disk drive

- M.2 NGFF SSD naar PCI-E 3.0 X16 High-Speed SSD (ashata = 5euro)
- need MVME (one notch) m2 card
- it shows up in BIOS / PCI devices
- on linux :#lsblk it should show up

2.3 Sata

- m2 to sata adapter case (aliexpress)
- cable female sata to female Slimline 13pin 7 + 6 (aliexpress)
- m2 sata disk 128G

2.4 Bootconfig

there is a - SAS controller - SATA controller (cdrom)

the controllor has a bootorder as well !!!! In order to boot from sata, the sata controller has to boot first!!!

**this is the way to boot an expensive hp server from a simple 2,5 inch sata disk (laptop 5V) ,using the onboard slimline cd-rom connector

CHAPTER

THREE

AVX

https://github.com/Mintplex-Labs/anything-llm/issues/1331

I ran into a problem using lancedb container that needs avx2. turns out that my old hp dl380p g8 with xeon e5-2650 v2 does not have AVX2....

AVX (Advanced Vector Extensions) and AVX2 are sets of CPU instructions designed to improve performance for tasks that involve heavy mathematical computations, like simulations, scientific calculations, or multimedia processing.

AVX (Advanced Vector Extensions):
Introduced with Intel's Sandy Bridge processors (2011). Supports 256-bit wide vector operations, allowing for parallel processing of multiple data points in a single instruction. This improves performance in applications like video processing, scientific simulations, and cryptographic tasks.

AVX2 (Advanced Vector Extensions 2):
Introduced with Intel's Haswell processors (2013). Expands on AVX by supporting more operations and improving the handling of integer operations. AVX2 also increases the vector size to 256 bits for integer data and introduces FMA (Fused Multiply-Add), which further boosts performance in floating-point operations.

Can the Xeon E5-2650 v2 Support AVX2?

No, the Intel Xeon E5-2650 v2 does not support AVX2. This processor, part of the Ivy Bridge-EP family, supports AVX but not AVX2, as AVX2 was introduced with the later Haswell architecture. Je zei: what do they mean : To resolve the issue all I had to do was update the docker run command with the lancedb_revert tag,

CHAPTER

FOUR

NVME

insert M2 into pci

check temperature : nvme smart-log /dev/nvme0 Smart Log for NVME device:nvme0 namespace-id:ffffffff critical_warning : 0 temperature : 33°C (306 Kelvin) available_spare : 100% available_spare_threshold : 10% percentage_used : 0% endurance group critical warning summary: 0 Data Units Read : 7,871 (4.03 GB) Data Units Written : 167,622 (85.82 GB) host_read_commands : 152,308 host_write_commands : 676,518 controller_busy_time : 1 power_cycles : 2 power_on_hours : 0 unsafe_shutdowns : 2 media_errors : 0 num_err_log_entries : 0 Warning Temperature Time : 0 Critical Composite Temperature Time : 0 Temperature Sensor 1 : 33°C (306 Kelvin) Thermal Management T1 Trans Count : 0 Thermal Management T2 Trans Count : 0 Thermal Management T1 Total Time : 0 Thermal Management T2 Total Time : 0

CHAPTER

FIVE

WHERE CAN I FIND MORE TEMPLATES?

- Use:

pveam update

- to update the container template database, then:

pveam available

CHAPTER

SIX

READING DATA FROM USB STICK

the USB stick is readable from the proxmox host:

(do a dmesg to get the device: in this case /dev/sdb1) mount /dev/sdb1 /usbdrive

6.1 create a mp on CT (container02)

/dev/mapper/pve-vm–102–disk–1 51290592 28 48652740 1% /container02mp

6.2 transfer to CT (name = container02)

on the host mkdir /drive_container02 mount /dev/mapper/pve-vm–102–disk–1 /drive_container02/

CHAPTER

SEVEN

RAMDISK

My dl380p gen8 has 256Gb of RAM

7.1 Can I use RAM as a disk?

mkdir /tmp/ramdisk chmod 777 /tmp/ramdisk mount -t tmpfs -o size=1024m myramdisk /tmp/ramdisk

or to get something extra…

sudo mount -t tmpfs -o size=10G myramdisk /tmp/ramdisk

7.2 speedtest

For Write: dd if=/dev/zero of=/dev/shm/ram bs=1048576 count=4096 oflag=nocache conv=fsync 4096+0 records in 4096+0 records out 4294967296 bytes (4.3 GB, 4.0 GiB) copied, 2.79948 s, 1.5 GB/s

or: dd if=/dev/zero of=/tmp/ramdisk/blok bs=1048576 count=1024 oflag=nocache conv=fsync 1024+0 records in 1024+0 records out 1073741824 bytes (1.1 GB, 1.0 GiB) copied, 0.560324 s, 1.9 GB/s

For Read:

dd if=/tmp/ramdisk/blok of=/dev/null bs=1048576 iflag=nocache,sync conv=nocreat

dd if=/tmp/ramdisk/blok of=/dev/null bs=1048576 iflag=nocache,sync conv=nocreat 1024+0 records in 1024+0 records out 1073741824 bytes (1.1 GB, 1.0 GiB) copied, 0.240446 s, 4.5 GB/s

---

modprobe zram echo 80G | tee /sys/block/zram0/disksize (80G ramdisk) mkfs.ext4 /dev/zram0 (make a filesystem) mkdir /RAM (create a mountingpoint) mount /dev/zram0 /RAM

now you can use the /RAM directory (which will be gone after poweroff)

CHAPTER

EIGHT

SHARING A DIRECTORY FOR LXC

on the host (pve node) create a shared directory (jan): (created on M2 storage) chown -R nobody:nogroup jan chmod 777 jan

root@pve:/etc/pve/lxc#

8.1 modify the lxc

create a directory /mnt/shared (which will be the mounting point)

8.2 on PVE modify the lxc config

CHAPTER

NINE

MOVING A LXC CONTAINER

I had a container on a logical volume SCSI and I wanted to move it to logical volume M2

Cloning?

The container shared a directory, and cloning and displacing would mess this up.

Solution: backup & restore from backup on other volume

CHAPTER

TEN

NFS NETWORK BETWEEN LINUX CONTAINERS

10.1 set up a bridge

A Linux bridge interface (commonly called vmbrX) is needed to connect guests to the underlying physical network. It can be thought of as a virtual switch which the guests and physical interfaces are connected to.

10.2 define an extra network interface in range 10.0.0.X

CHAPTER

ELEVEN

NFS HOST AND CLIENT SETUP IN PROXMOX

This guide will explain how to set up an NFS (Network File Sharing) server and add it as a remote storage in Proxmox.

1. **Install NFS Server**

   First, log in to the LXC container or the machine where the NFS server will be hosted. Then update the package list and install NFS:

   ```
   apt-get update
   apt-get install sudo -y
   sudo apt install nfs-kernel-server
   ```

   Once installed, create a shared folder:

   ```
   sudo mkdir /home/sharedfolder
   sudo chmod 777 /home/sharedfolder
   ```

   Next, edit the `/etc/exports` file to configure the shared directory for export:

   ```
   sudo nano /etc/exports
   ```

   Add the following line (adjust the IP address and folder accordingly):

   ```
   /home/sharedfolder 192.168.1.0/24(rw,sync,no_subtree_check)
   ```

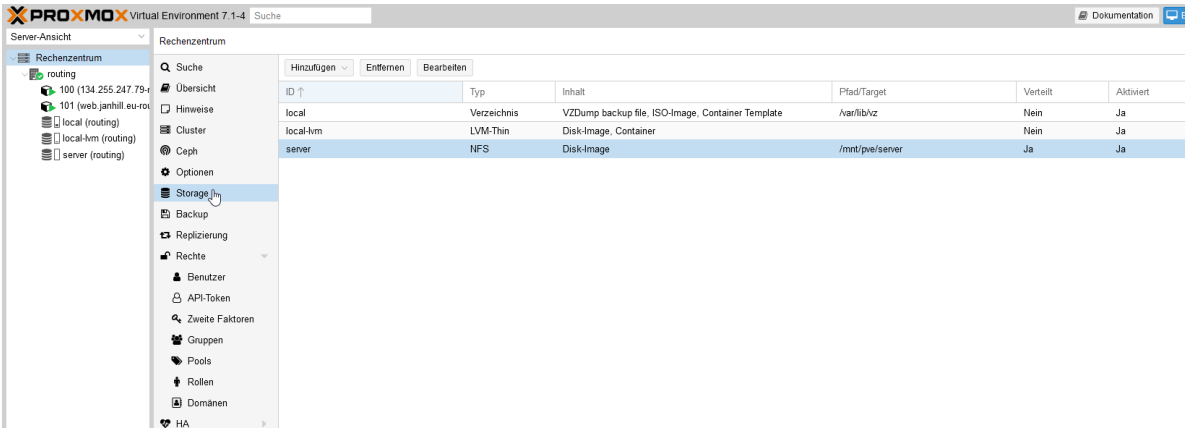
   Save the file and restart the NFS server:

   ```
   sudo exportfs -ra
   sudo systemctl restart nfs-kernel-server
   ```

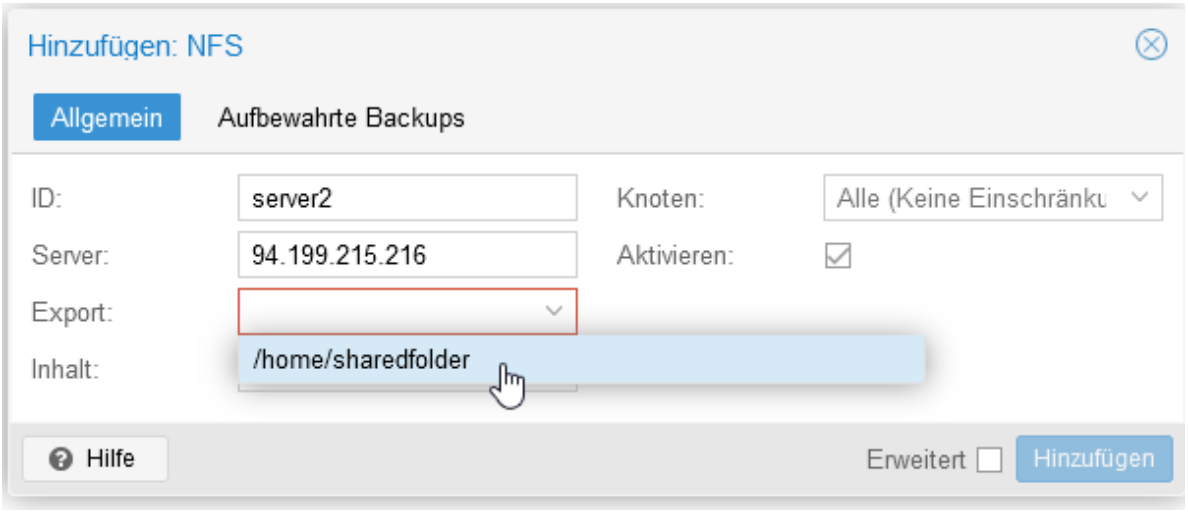
2. **Configure NFS Storage in Proxmox**

   Now log into your Proxmox host (the machine that will receive the NFS storage) and navigate to:

   Datacenter > *Storage* > *Add* > *NFS*



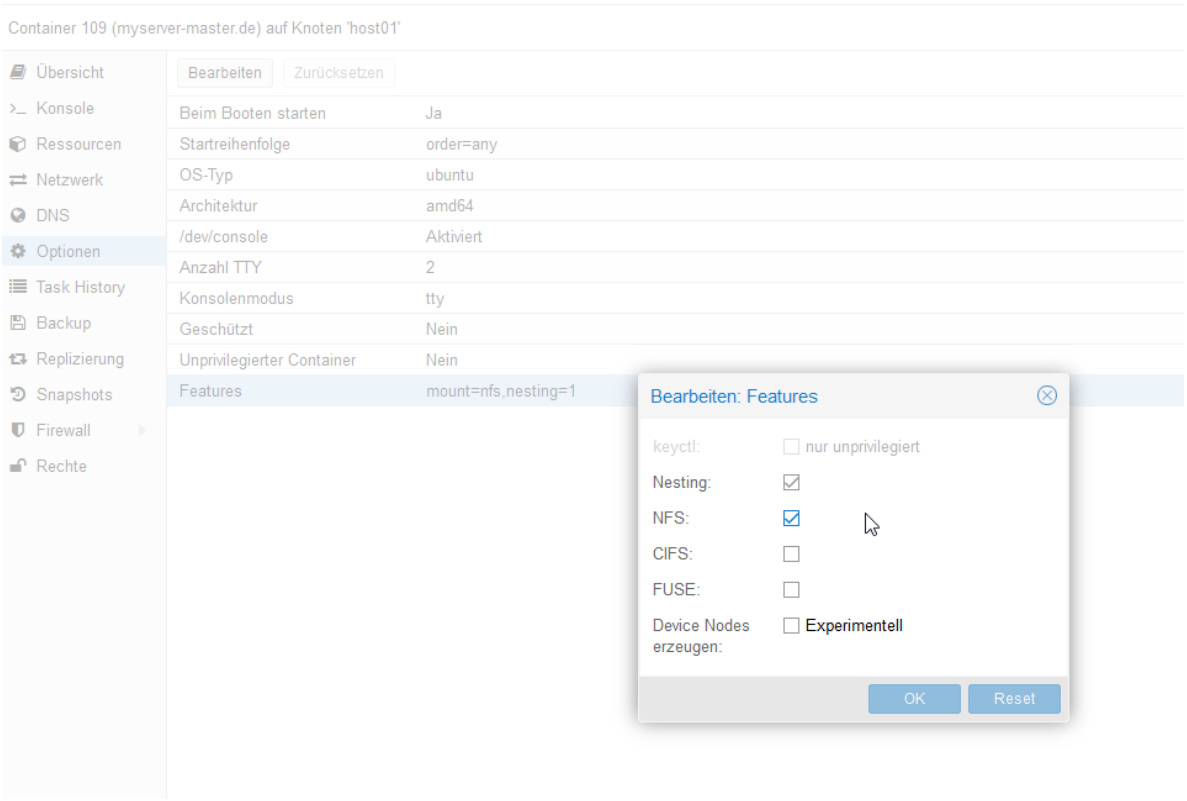
Here, enter the NFS server's IP address and select the shared directory.



Choose the desired contents for the storage (ISO images, containers, backups, etc.) and click *Add*.

3. **Set Permissions on LXC Container (If Applicable)**

   If the NFS share will be used in an LXC container, ensure that permissions for NFS usage are set correctly:



Check the *NFS* box under *Options* for the LXC container.

That's it! You have successfully set up an NFS server and added it to Proxmox as remote storage.

CHAPTER

TWELVE

NFS CLIENT

apt install nfs-common

mount -t nfs 10.0.0.104:/share /mnt/nfson104

12.1 under node pve

watch out for features

CHAPTER

THIRTEEN

PORTAINER

13.1 using a script

use a template for the linux container

a script from : https://raw.githubusercontent.com/tteck/Proxmox/refs/heads/main/install/docker-install.sh

systemctl start docker systemctl status docker

https://192.168.0.182:9443 (your IP)

13.2 by hand

- create CT ubuntu22 with template
- apt update
- sudo apt install docker.io -y
- sudo systemctl status docker
- sudo usermod -aG docker $USER (add current logged on user to docker group)
- docker pull portainer/portainer-ce:latest
- docker run -d -p 9000:9000 –restart always -v /var/run/docker.sock:/var/run/docker.sock portainer/portainer-ce:latest

13.3 exporting & importing

this seems to work between systems:

(origin) sudo docker save ollama/ollama:latest > my-ollama.tar (target) sudo docker load < my-ollama.tar

CHAPTER

FOURTEEN

SERVER FOR BLENDER

Although the hp dl380 was 5 times faster than my i7 laptop, it is still slow, takes half an hour to render a picture.

One can adjust setting within blender to speed up things a bit, but still …

14.1 howto render?

- download blender 4.2

cd blender-4.2.0-linux-x64/

./blender -b /home/naj/misvormde-donut1.blend -E CYCLES -f 1

14.2 faster / less good

./blender -b /home/naj/misvormde-donut1.blend -E BLENDER_EEVEE_NEXT -f 1

14.3 at the movies

./blender -b /home/naj/misvormde-donut1.blend -E BLENDER_EEVEE_NEXT -s 10 -e 500 -t 2 -a ./blender -b /home/naj/misvormde-donut1.blend -E BLENDER_EEVEE_NEXT -s 1 -e 100 -t 2 -a

CHAPTER

FIFTEEN

OLLAMA INSTALL AND USE

curl -fsSL https://ollama.com/install.sh | sh

15.1 using a container

ollama-model-gemma2 was mounted using a volume and the image exported

sudo docker import ollama.tar ollama:latest

CHAPTER

SIXTEEN

DOCKER

using ubuntu 22, I found out that the docker that came with it does not work like it should

- uninstall docker

docker version Client: Docker Engine - Community

Version: 27.3.1 API version: 1.47 Go version: go1.22.7 Git commit: ce12230 Built: Fri Sep 20 11:41:00 2024 OS/Arch: linux/amd64

CHAPTER

SEVENTEEN

RUNNING OPENWEBUI

- running it from within portainer did not allow to change the host
- command prompt

sudo docker run -d -p 3000:8080 –add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data –name open-webui –restart always ghcr.io/open-webui/open-webui:main

CHAPTER

EIGHTEEN

COPY DATA TO CONTAINER

docker cp manifests/ ollama:/root/.ollama/models Successfully copied 12.8kB to ollama:/root/.ollama/models

CHAPTER

NINETEEN

CONFIGURE ELASTICVIEW

19.1 elasticsearch password

connect with term to container: bin/elasticsearch-reset-password -u elastic bin/elasticsearch-reset-password -u elastic -i (this allows for setting it yourself)

19.2 testing elasticview connection

curl -X GET -k -u elastic:ebktuBhBHtE7N+JeBbIV “https://192.168.0.121:9200/_cluster/health/?pretty”

{
“cluster_name” : “docker-cluster”, “status” : “green”, “timed_out” : false, “number_of_nodes” : 1, “number_of_data_nodes” : 1, “active_primary_shards” : 1, “active_shards” : 1, “relocating_shards” : 0, “initializing_shards” : 0, “unassigned_shards” : 0, “delayed_unassigned_shards” : 0, “number_of_pending_tasks” : 0, “number_of_in_flight_fetch” : 0, “task_max_waiting_in_queue_millis” : 0, “active_shards_percent_as_number” : 100.0

}

19.3 getting website certificate

connect to http://192.168.0.121

firefox/settings/privicy&security/certificates to add exception

CHAPTER

TWENTY

MOUNTING

mkdir /SCSIdata for the data on the scsidisk /dev/sdb mkdir /M2data for the data on the M2 disk /dev/nvme0n1p2

mount /dev/sdb /SCSIdata

mount /dev/nvme0n1p2 /M2data

CHAPTER

TWENTYONE

EDIT /ETC/FSTAB

/dev/nvme0n1p2 /M2data ext4 defaults 0 2 /dev/sdb /SCSIdata ext4 defaults 0 2

CHAPTER

TWENTYTWO

BACKUP PVE

Proxmox node now start from a sata m2 in the CDrom slot

A tar copy is made to M2data from /etc directory

In case of crash : reinstall proxmox on sata disk en copy backup.tar to /etc

CHAPTER

TWENTYTHREE

RAID

previously 2 600GB SAS disks were in RAID1 in the same volumegroup.

I have no spare disk, nor do I want to spend money on old tech.

Reconfigure : each disk is within own volume group, no more RAID, more (free) space. Tool would not let me otherwise.

23.1 No more raid :

machine on one disk are backupped to other disk, and vice versa one disk remains VG (volume group) within proxmox : used for LXC en VM images, backup other disk contains ext4 filesystem and is a directory

CHAPTER

TWENTYFOUR

MELBORP SERVER

24.1 solution for problem running pgadmin container and nginx

Configure Nginx: Create a new configuration file for your domain in /etc/nginx/sites-available/melborp.solutions:

nginx

server {
listen 80; server_name www.melborp.solutions;

location / {
proxy_pass http://localhost:8888; # Forward traffic to the pgAdmin container proxy_set_header Host $host; proxy_set_header X-Real-IP $remote_addr; proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; proxy_set_header X-Forwarded-Proto $scheme;

}

}

CHAPTER

TWENTYFIVE

SETTING UP NVIDIA TESLA P100

the P100 was choses for balance between performance and price (second hand ebay)

25.1 Aliexpress

PCIE 4.0 3.0 16X Riser Kabel 90/180 Graden Mount Video Grafisch

Gpu 10pin Naar 1x8pin Dual 8(6 + 2)Pin Power Adapter Kabel Voor Hp Dl380 Gen8 Gen9 Server

25.2 Cutting the Riser

a special Riser for the HP Proliant gen8 is a good idea because it has a PCIE 16x in the middle which allows for the GPU to be plugged in. (hard to find and expensive)

I choose the cheap option (evidently) and cut a hole in a riser to extend with a riser cable into this riser. Some more cutting had to be done because the GPU is too long.

For the gen9 there a cheaper riser card options, so no cutting needed…

25.3 Configure the BIOS

(changed IRQ for network card to 11, since conflict)

enter BIOS (F9) and on main bios screen / "Service options" menu item. Under this, enable "PCI Express 64-bit BAR

25.4 configure proxmox host

PROXMOX PCI-E GPU passthru

When running proxmox on this hardware, there is more config needed to enable passthru of this GPU to VM. Enable IOMMU

25.5 the idea

the idea is to get nvidia to work on the proxmox host (there will be kernel modules)

for the LXC machines there is no need for kernel drivers!! since already on the host

25.6 Download NVIDIA drivers

25.7 adapt lxc.conf

/etc/pve/lxc/105.conf

Append these values with the IDs you noted earlier to your file like so. Note the placement of the 195, 234 and 509. This is for a SINGLE gpu also, if you have multiple add additional

25.8 copy NVIDIA driver to LXC

25.9 Install the NVIDIA Container Toolkit

25.10 install ollama docker container for GPU usage

#in portainer you still need to start ollama this way:

CHAPTER

TWENTYSIX

LAPTOP (OR PC) MODS

/etc/hosts

10.10.10.50 nginx.example.com 10.10.10.50 registry.example.com

ip route add 10.10.10.0/24 via 192.168.0.251

sudo cp ca.crt /usr/local/share/ca-certificates/ca.crt sudo update-ca-certificates Updating certificates in /etc/ssl/certs...

26.1 make route permanent

```
sudo nano /etc/netplan/01-netcfg.yaml

network:
version: 2
ethernets:
  wlp0s20f3:
    addresses:
      - 192.168.0.103/24
    gateway4: 192.168.0.1  # Replace with your actual default gateway
    routes:
      - to: 10.10.10.0/24
        via: 192.168.0.251

  sudo netplan apply
```

CHAPTER

TWENTYSEVEN

PROXMOX

remove firewall from talos worker node

```
root@pve:~# qm set 109 -net0 virtio,bridge=vmbr0,firewall=0
update VM 109: -net0 virtio,bridge=vmbr0,firewall=0
```

27.1 setting up an extra network bridge vmbr1

on lxc dedicated machine setup dhcp and routing

```
apt install dnsmasq -y
apt install iptables-persistent -y

vi /etc/dnsmasq.conf
interface=eth0
dhcp-range=10.10.10.100,10.10.10.200,12h  # DHCP range for Talos nodes
dhcp-option=3,10.10.10.2                  # Gateway (this machine's eth0 IP)
dhcp-option=6,192.168.0.1                 # DNS (your home router's DNS)

systemctl restart dnsmasq

echo 1 > /proc/sys/net/ipv4/ip_forward

iptables -t nat -L -v
ip route del default via 10.10.10.1 dev eth0
ip route replace default via 192.168.0.1 dev eth1 metric 100

iptables -t nat -A POSTROUTING -s 10.10.10.0/24 -o eth1 -j MASQUERADE
ip route del default via 10.10.10.1 dev eth0
ip route replace default via 192.168.0.1 dev eth1 metric 100



vi /etc/netplan/01-netcfg.yaml
```

```
network:
  version: 2
  ethernets:
    eth0:
```

(continues on next page)

(continued from previous page)

```
      dhcp4: true
      # Prevent DHCP from setting a default gateway if it conflicts
      dhcp4-overrides:
        use-routes: true
        use-dns: true
        route-metric: 2000  # High metric to prioritize eth1's default route
    eth1:
      dhcp4: false
      addresses:
        - 192.168.0.x/24  # Replace with your server's IP on this subnet
      routes:
        - to: 0.0.0.0/0
          via: 192.168.0.1
          metric: 100
        - to: 0.0.0.0/0
          via: 192.168.0.1
          metric: 1024
        - to: 192.168.0.0/24
          via: 0.0.0.0
          metric: 1024
        - to: 192.168.0.1
          via: 0.0.0.0
          metric: 1024
        - to: <gent.dnscache01-ip>
          via: 192.168.0.1
          metric: 1024
        - to: <gent.dnscache02-ip>
          via: 192.168.0.1
          metric: 1024
```

```
# Apply the netplan configuration
sudo netplan generate
sudo netplan apply

# Check the routing table
ip route show

# Check iptables rules
iptables -t nat -L -v

# Check dnsmasq status
systemctl status dnsmasq

# Check if the DHCP server is running and listening on the correct interface
sudo systemctl status dnsmasq

# Restart dnsmasq to apply changes
sudo systemctl restart dnsmasq


netplan apply
```

27.2 Kernel IP routing table

27.3 using the nodeport

192.168.0.251:30743

on my router/dhcp on 10.10.10.2 route port to cluster node IP

iptables -t nat -A PREROUTING -i eth1 -p tcp –dport 30743 -j DNAT –to-destination 10.10.10.118:30743

so running nginx on kubernetes on 10.10.10.255 network is accessible from the outside

27.4 using the IP address

traefik LoadBalancer 10.102.122.212 10.10.10.50 80:32178/TCP,443:32318/TCP 75m app.kubernetes.io/instance=traefik-default,app.kubernetes.io/name=traefik

So now I have to figure out how I can reach 10.10.10.50 from my 192.168.X.X network

on the kubernetes cluster, traefik has been deployed as well as metallb. iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -d 10.10.10.0/24 -j MASQUERADE sh -c “iptables-save > /etc/iptables/rules.v4”

this has been added to th dnsmasq.conf

Listen on the 192.168.0.251 interface interface=eth1 # Replace with your 192.168.0.251 interface (check with *ip a*) listen-address=192.168.0.251

Forward other queries to upstream DNS (e.g., Google DNS) server=8.8.8.8 server=8.8.4.4

Optional: If LXC is your DHCP server, ensure DNS is offered dhcp-option=6,192.168.0.251 # Tells DHCP clients to use this as DNS

27.5 modify dns config on laptop

/etc/resolv.conf

add : nameserver 192.168.0.251

27.6 access http://nginx.example.com/ on talos within 10.10.10.X from 192.168.X.X

(configure metallb, traefik, nginx)

on laptop /etc/hosts : 10.10.10.50 nginx.example.com

on dhcp server (10.10.10.2)

iptables -A FORWARD -s 192.168.0.0/24 -d 10.10.10.0/24 -j ACCEPT iptables -A FORWARD -s 10.10.10.0/24 -d 192.168.0.0/24 -j ACCEPT

```
# Generated by iptables-save v1.8.7 on Thu Apr 10 13:32:59 2025
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A FORWARD -s 192.168.0.0/24 -d 10.10.10.0/24 -j ACCEPT
-A FORWARD -s 10.10.10.0/24 -d 192.168.0.0/24 -j ACCEPT
COMMIT
# Completed on Thu Apr 10 13:32:59 2025
# Generated by iptables-save v1.8.7 on Thu Apr 10 13:32:59 2025
*nat
:PREROUTING ACCEPT [6847:1975161]
:INPUT ACCEPT [158:15156]
:OUTPUT ACCEPT [25:2590]
:POSTROUTING ACCEPT [25:2590]
-A PREROUTING -i eth1 -p tcp -m tcp --dport 30743 -j DNAT --to-destination 10.10.10.
↪118:30743
-A POSTROUTING -s 10.10.10.0/24 -o eth1 -j MASQUERADE
-A POSTROUTING -s 10.10.10.0/24 -o eth1 -j MASQUERADE
-A POSTROUTING -s 10.10.10.0/24 -o eth1 -j MASQUERADE
-A POSTROUTING -s 192.168.0.0/24 -d 10.10.10.0/24 -j MASQUERADE
COMMIT
# Completed on Thu Apr 10 13:32:59 2025
# Check the iptables rules
iptables -t nat -L -v
iptables -L -v

# Check the routing table
ip route show

# Check the network interfaces
ip a
```

CHAPTER

TWENTYEIGHT

TALOS

Talos is a modern OS for Kubernetes. It is designed to be secure, immutable, and minimal. Talos is a self-hosted Kubernetes distribution that runs on bare metal or virtualized infrastructure. Talos is designed to be managed by a central Kubernetes control plane, which can be hosted on the same cluster or on a separate cluster.

talosctl config add my-cluster –endpoints 192.168.0.242

talosctl config info

talosctl config endpoint 192.168.0.242

talosctl gen config my-cluster https://192.168.0.242:6443 –output-dir ./talos-config –force

28.1 new install talos

https://www.talos.dev/v1.9/talos-guides/install/virtualized-platforms/proxmox/

> talosctl gen config my-cluster https://192.168.0.218:6443 talosctl -n 192.168.0.169 get disks –insecure (check disks) talosctl config endpoint 192.168.0.218 talosctl config node 192.168.0.218
>
> talosctl apply-config –insecure –nodes 192.168.0.218 –file controlplane.yaml
>
> talosctl bootstrap talosctl kubeconfig . (retrieve kubeconfig) talosctl –nodes 192.168.0.218 version (verify)
>
> export KUBECONFIG=./talos-config/kubeconfig
>
> > kubectl get nodes kubectl get pods -n kube-system kubectl get pods -n kube-system -o wide
>
> kubectl describe pod my-postgres-postgresql-0 (is very useful in case the pod does get deployed
>
> https://factory.talos.dev/ (create your custom image)

```
talosctl upgrade --nodes 10.10.10.178 --image  factory.talos.dev/installer/
↪c9078f9419961640c712a8bf2bb9174933dfcf1da383fd8ea2b7dc21493f8bac:v1.9.5
```

watching nodes: [10.10.10.178]
talosctl get extensions –nodes 10.10.10.178

NODE NAMESPACE TYPE ID VERSION NAME VERSION 10.10.10.178 runtime ExtensionStatus 0 1 iscsi-tools v0.1.6 10.10.10.178 runtime ExtensionStatus 1 1 schematic c9078f9419961640c712a8bf2bb9174933dfcf1da383fd8ea2b7dc21493f8bac

28.2 adding worker nodes

Since “longhorn” stores data on more than one node, we need to add more nodes to the cluster.

> talosctl apply-config –insecure –nodes 10.10.10.166 –file worker.yaml talosctl apply-config –insecure –nodes 10.10.10.173 –file worker.yaml

kubectl get nodes -o wide NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME talos-2ho-roe Ready <none> 113s v1.32.3 10.10.10.173 <none> Talos (v1.9.5) 6.12.18-talos containerd://2.0.3 talos-swn-isw Ready control-plane 31d v1.32.3 10.10.10.118 <none> Talos (v1.9.5) 6.12.18-talos containerd://2.0.3 talos-v1x-9s4 Ready <none> 2m18s v1.32.3 10.10.10.166 <none> Talos (v1.9.5) 6.12.18-talos containerd://2.0.3 talos-y7t-8ll Ready worker 29d v1.32.3 10.10.10.178 <none> Talos (v1.9.5) 6.12.18-talos containerd://2.0.3

28.3 label nodes

> kubectl label nodes talos-v1x-9s4 node-role.kubernetes.io/worker=”” kubectl label nodes talos-2ho-roe node-role.kubernetes.io/worker=””

CHAPTER

TWENTYNINE

KERNEL IP ROUTING TABLE

The following table represents the kernel IP routing table:

Table 1: Kernel IP Routing Table

Destination	Gateway	Genmask	Flags	Met-ric	Ref	Use	Iface
default	192.168.0.1	0.0.0.0	UG	100	0	0	eth1
default	192.168.0.1	0.0.0.0	UG	1024	0	0	eth1
10.10.10.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.0.0	0.0.0.0	255.255.255.0	U	1024	0	0	eth1
192.168.0.1	0.0.0.0	255.255.255.255	UH	1024	0	0	eth1
gent.dnscache01	192.168.0.1	255.255.255.255	UGH	1024	0	0	eth1
gent.dnscache02	192.168.0.1	255.255.255.255	UGH	1024	0	0	eth1

CHAPTER

THIRTY

SDA (SOFTWARE DEFINED ARCHITECTURE)

30.1 Physical Infrastructure

30.2 Proxmox Server Specifications

The foundation of the architecture is a physical server running Proxmox VE hypervisor.

Component	Description
Hypervisor	Proxmox Virtual Environment (VE)
Virtual Machines	Multiple Talos OS nodes forming a Kubernetes cluster
Containers	LXC container serving as a router

30.3 Virtual Machine Configuration

Talos OS Nodes

The cluster consists of multiple Talos OS nodes, with dedicated roles:

- **Control Plane Node(s)**: Manages the Kubernetes control plane
- **Worker Nodes**: Runs application workloads

```
# Example Talos configuration structure (simplified)
machine:
  type: controlplane  # or worker
  network:
    hostname: talos-node-1
  kubernetes:
    version: v1.26.0
```

30.4 Networking Architecture

30.4.1 Network Components

Component	Function
Proxmox Virtual Bridge	Creates isolated network segments for VMs and containers
LXC Router	Routes traffic between internal and external networks
Kubernetes Overlay Network	Enables pod-to-pod communication (Cilium, Flannel, etc.)

30.5 Control & Automation

30.5.1 API Management Layer

This architecture leverages multiple declarative APIs for infrastructure management:

API	Responsibility
Proxmox API	Manages physical resources, VMs, and containers
Talos API	Provides declarative OS configuration and maintenance
Kubernetes API	Orchestrates applications and services

30.6 Benefits of This Architecture

- **Immutable Infrastructure**: Talos OS provides an immutable, declarative operating system
- **High Availability**: Kubernetes manages service availability and distribution
- **Resource Efficiency**: Consolidates multiple services on a single physical server
- **Isolation**: Separate network segments and container boundaries
- **Automation**: API-driven management at all levels

CHAPTER

THIRTYONE

INDICES AND TABLES

- genindex
- modindex
- search