# Maschinen

Doku über die Maschinen (virtuell und physikalisch) des PING e.V.

- vserver.ping.de
- Übersicht über die Softwarestände der Produktivsysteme
- Hetzner-Server
- Mailserver
- KI-Server

# vserver.ping.de

# Was ist LXC und warum machen wir damit VServer

LXC steht für Linux-Containers und ist eine Art virtueller Server für Linux. Dabei nutzen alle virtuellen Server-Instanzen den Kernel der Wirtsystems mit. Implementiert werden die Container in Linux über die Namespace Erweiterungen, die dafür sorgen, dass Prozesse in eigene Adressräume isoliert werden können. Da dies nicht nur Prozesse und Speicher, sondern auch Netzwerke umfasst, können hiermit vollständige virtuelle Server implementiert werde.

Bislang wurde bei PING die ältere Linux-VServer-Implentierung eingesetzt, die einen eigenen Kernel-Patch und Userspace-Tools implementiert. Bis zur Version Squeeze von Debian wurde dabei vom Debian-Kernel-Team ein entsprechend gepatchter Kernel zur Verfügung gestellt. Mit Release von Wheezy wurde dies allerdings zu Gunsten der weiter fortgeschrittenen Entwicklung der Linux-Containers entfernt.

Für Linux-Containers sind einige Optionen zu Namespaces zu aktivieren, was aber bei den Kernel-Images von Debian bereits der Fall ist. Es ist somit nur noch das Paket *lxc* zu installieren, welches die Userspace-Tools zu Erzeugung und Verwaltung der Container enthält.

# Probleme

LXC ist noch in der Entwicklung, früher waren LXC-Container nicht ausbruchsicher, dies ist aber seit Debian Jessie nicht mehr der Fall siehe https://wiki.debian.org/LXC

# Installation des Hosts

Der Anleitung im Debian Wiki folgend, muss zunächst das LXC-Paket installiert werden:

```
root@containers:~# apt-get install lxc
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
The following additional packages will be installed:
  arch-test bridge-utils busybox-static cloud-image-utils debootstrap distro-info dns-root-data
Suggested packages:
  cloud-utils-euca ubuntu-archive-keyring squid-deb-proxy-client shunit2 wodim cdrkit-doc btrfs-
The following NEW packages will be installed:
  arch-test bridge-utils busybox-static cloud-image-utils debootstrap distro-info dns-root-data
0 upgraded, 23 newly installed, 0 to remove and 0 not upgraded.
Need to get 8030 kB of archives.
After this operation, 34.4 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

# Vorbereitung der unprivilegierten Root-Container

Da die als VServer gedachten Container von Root während des Startes des Systems erzeugt werden, diese aber unprivilegiert sein sollen, muss ein uid- und gid-Bereich dem Nutzer Root zugewiesen werden (vergl. https://linuxcontainers.org/lxc/getting-started/#creating-unprivileged-containers-as-root):

```
root@containers:/etc# git diff
diff --git a/lxc/default.conf b/lxc/default.conf
index d2d1d51..30231e3 100644
--- a/lxc/default.conf
+++ b/lxc/default.conf
@@ -1,3 +1,5 @@
 lxc.net.0.type = empty
 lxc.apparmor.profile = generated
 lxc.apparmor.allow_nesting = 1
+lxc.idmap = u 0 100000 65536
+lxc.idmap = g 0 100000 65536
diff --git a/subgid b/subgid
index dc7d014..44be50f 100644
--- a/subgid
+++ b/subgid
@@ -1,2 +1 @@
-dh:100000:65536
+root:100000:65536
diff --git a/subuid b/subuid
index dc7d014..44be50f 100644
```

```
--- a/subuid
+++ b/subuid
@@ -1,2 +1 @@
-dh:100000:65536
+root:100000:65536
```

# Netzwerk

Damit die LXC-Container auf das Netwerk zugreifen können, müssen deren virtuellen Netzwerkkarten an das `lxcbr0` angebunden werden. Da `vserver.ping.de` selbst eine virtuelle Maschine ist und die IP-Netze außerhalb der VM ankommen, bzw. dorthin geroutet werden, muss für die Container einer Bridge eingerichtet werden, in der die an vserver.ping.de gerouteten Netze verwendet:

```
auto lxcbr0
iface lxcbr0 inet static
        bridge_ports none
        bridge_stp off
        bridge_fd 0
        address 176.9.167.1
        netmask 255.255.255.240

iface lxcbr0 inet6 static
        bridge_ports none
        bridge_stp off
        bridge_fd 0
        address 2a01:4f8:262:445e:300::1/72
```

Damit LXC für die erzeugten Container auch ein Interface im Host erzeugt und dies der Bridge hinzufügt, muss `/etc/lxc/default.conf` angepasst werden:

```
diff --git a/lxc/default.conf b/lxc/default.conf
index 30231e3..c6bd25e 100644
--- a/lxc/default.conf
+++ b/lxc/default.conf
@@ -1,4 +1,6 @@
-lxc.net.0.type = empty
+lxc.net.0.type = veth
```

```
+lxc.net.0.link = lxcbr0
+lxc.net.0.flags = up
 lxc.apparmor.profile = generated
 lxc.apparmor.allow_nesting = 1
 lxc.idmap = u 0 100000 65536
 lxc.idmap = g 0 100000 65536
 lxc.cgroup.devices.allow = c 10:200 rwm
```

# Container erstellen

Container speichern ihr Dateisystem unter `/var/lib/lxc/<name>/rootfs`. Aus diesem Grund sollte hier ein Dateisystem für den Container eingehängt werden (z.B. ein LV aus dem Dom0 durchgereicht).

Auf vserver.ping.de gibt es ein LVM-Volume `vserver` in dem für jeden VServer ein LV mit einem Dateisystem angelegt und eingehängt wird.

```
root@vserver:~# lvcreate -L 70G -n jmp100.prima.de vserver
  Logical volume "jmp100.prima.de" created.

root@vserver:~# mkfs.ext4 /dev/vserver/jmp100.prima.de
mke2fs 1.47.2 (1-Jan-2025)
Discarding device blocks: done
Creating filesystem with 18350080 4k blocks and 4587520 inodes
Filesystem UUID: eb9c3ac5-baf0-4dba-92d0-d3ab68097894
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624, 11239424

Allocating group tables: done
Writing inode tables: done
Creating journal (131072 blocks): done
Writing superblocks and filesystem accounting information: done

root@vserver:~# mkdir /var/lib/lxc/jmp100.prima.de

root@vserver:~# echo -e
"/dev/vserver/jmp100.prima.de\t/var/lib/lxc/jmp100.prima.de\text4\tdefaults\t0\t2" >>
```

```
/etc/fstab

root@vserver:~# systemctl daemon-reload

root@vserver:~# mount /var/lib/lxc/jmp100.prima.de
```

Das

Da die genutzten Container unprivilegiert sind, muss das D`ownload`-Template genutzt werden. Hier ist dann die `Distribution` (z.B.: `Debian`), das `Release` (z.B.: `buster`) und die `Architektur` (z.B.: `amd64`) angegeben werden.

```
root@vserver:~# lxc-create -n jmp100.prima.de -t download
Downloading the image index

---
DIST            RELEASE     ARCH   VARIANT  BUILD
---
almalinux       10          amd64  default  20260123_23:25
almalinux       10          arm64  default  20260123_23:29
almalinux       8           amd64  default  20260123_23:23
almalinux       8           arm64  default  20260123_23:27
almalinux       9           amd64  default  20260123_23:25
almalinux       9           arm64  default  20260123_23:28
alpine          3.20        amd64  default  20260124_13:00
alpine          3.20        arm64  default  20260124_13:01
alpine          3.20        armhf  default  20260124_13:00
alpine          3.20        riscv64  default  20260124_13:07
alpine          3.21        amd64  default  20260124_13:00
alpine          3.21        arm64  default  20260124_13:01
alpine          3.21        armhf  default  20260124_13:02
alpine          3.21        riscv64  default  20260124_13:02
alpine          3.22        amd64  default  20260124_13:00
alpine          3.22        arm64  default  20260124_13:02
alpine          3.22        armhf  default  20260124_13:01
alpine          3.22        riscv64  default  20260124_13:02
alpine          3.23        amd64  default  20260124_13:00
alpine          3.23        arm64  default  20260124_13:00
alpine          3.23        armhf  default  20260124_13:00
alpine          3.23        riscv64  default  20260124_13:03
```

```
alpine          edge        amd64  default  20260124_13:00
alpine          edge        arm64  default  20260124_13:00
alpine          edge        armhf  default  20260124_13:01
alpine          edge        riscv64  default  20260124_13:06
alt             Sisyphus    amd64  default  20260124_03:08
alt             Sisyphus    arm64  default  20260124_03:22
alt             p11         amd64  default  20260124_03:40
alt             p11         arm64  default  20260124_03:17
amazonlinux     2           amd64  default  20260124_05:09
amazonlinux     2           arm64  default  20260124_05:09
amazonlinux     2023        amd64  default  20260124_05:09
archlinux       current     amd64  default  20260124_04:18
archlinux       current     arm64  default  20260124_04:18
archlinux       current     riscv64  default  20260124_04:18
busybox         1.36.1      amd64  default  20260124_06:00
busybox         1.36.1      arm64  default  20260124_06:00
centos          10-Stream   amd64  default  20260124_07:08
centos          10-Stream   arm64  default  20260124_07:35
centos          9-Stream    amd64  default  20260124_07:35
centos          9-Stream    arm64  default  20260124_07:08
debian          bookworm    amd64  default  20260124_05:24
debian          bookworm    arm64  default  20260124_05:24
debian          bookworm    armhf  default  20260124_05:24
debian          bullseye    amd64  default  20260124_05:24
debian          bullseye    arm64  default  20260124_05:24
debian          bullseye    armhf  default  20260124_05:39
debian          forky       amd64  default  20260124_05:24
debian          forky       arm64  default  20260124_05:24
debian          forky       armhf  default  20260124_05:24
debian          forky       riscv64  default  20260124_05:40
debian          trixie      amd64  default  20260124_05:24
debian          trixie      arm64  default  20260124_05:24
debian          trixie      armhf  default  20260124_05:36
debian          trixie      riscv64  default  20260124_05:24
devuan          chimaera    amd64  default  20260124_11:50
devuan          chimaera    arm64  default  20260124_11:50
devuan          daedalus    amd64  default  20260124_11:50
devuan          daedalus    arm64  default  20260124_11:50
devuan          excalibur   amd64  default  20260124_11:50
devuan          excalibur   arm64  default  20260124_11:50
```

```
fedora          41          amd64  default  20260124_00:01
fedora          41          arm64  default  20260123_23:38
fedora          42          amd64  default  20260123_23:40
fedora          42          arm64  default  20260124_00:09
fedora          43          amd64  default  20260124_00:01
fedora          43          arm64  default  20260124_00:39
kali            current     amd64  default  20260124_17:14
kali            current     arm64  default  20260124_17:14
mint            ulyana      amd64  default  20260124_08:51
mint            ulyssa      amd64  default  20260124_08:51
mint            uma         amd64  default  20260124_08:51
mint            una         amd64  default  20260124_08:51
mint            vanessa     amd64  default  20260124_08:51
mint            vera        amd64  default  20260124_08:51
mint            victoria    amd64  default  20260124_08:51
mint            virginia    amd64  default  20260124_08:51
mint            wilma       amd64  default  20260124_08:51
nixos           25.11       amd64  default  20260124_01:00
nixos           25.11       arm64  default  20260124_01:50
nixos           unstable    amd64  default  20260124_01:02
nixos           unstable    arm64  default  20260124_01:50
openeuler       20.03       amd64  default  20260120_15:48
openeuler       20.03       arm64  default  20260120_15:48
openeuler       22.03       amd64  default  20260120_15:48
openeuler       22.03       arm64  default  20260120_15:48
openeuler       24.03       amd64  default  20260120_15:48
openeuler       24.03       arm64  default  20260120_15:48
openeuler       25.03       amd64  default  20260120_15:48
openeuler       25.03       arm64  default  20260120_15:48
openeuler       25.09       amd64  default  20260120_15:48
openeuler       25.09       arm64  default  20260120_15:48
opensuse        15.6        amd64  default  20260124_04:20
opensuse        15.6        arm64  default  20260124_04:20
opensuse        16.0        amd64  default  20260124_04:20
opensuse        16.0        arm64  default  20260124_04:20
opensuse        tumbleweed  amd64  default  20260124_04:20
opensuse        tumbleweed  arm64  default  20260124_04:20
openwrt         23.05       amd64  default  20260124_11:57
openwrt         23.05       arm64  default  20260124_11:57
openwrt         24.10       amd64  default  20260124_11:57
```

```
openwrt          24.10       arm64  default  20260124_11:57
openwrt          25.12       amd64  default  20260124_11:57
openwrt          25.12       arm64  default  20260124_11:57
openwrt          snapshot    amd64  default  20260124_11:57
openwrt          snapshot    arm64  default  20260124_11:57
oracle           7           amd64  default  20260124_07:46
oracle           7           arm64  default  20260124_08:10
oracle           8           amd64  default  20260124_07:46
oracle           8           arm64  default  20260124_08:12
oracle           9           amd64  default  20260124_07:46
oracle           9           arm64  default  20260124_08:09
plamo            8.x         amd64  default  20260124_03:08
rockylinux       10          amd64  default  20260124_03:08
rockylinux       10          arm64  default  20260124_03:08
rockylinux       8           amd64  default  20260124_03:08
rockylinux       8           arm64  default  20260124_03:08
rockylinux       9           amd64  default  20260124_03:08
rockylinux       9           arm64  default  20260124_03:14
slackware        15.0        amd64  default  20260123_23:27
slackware        current     amd64  default  20260123_23:26
springdalelinux  7           amd64  default  20260124_06:38
springdalelinux  8           amd64  default  20260124_06:38
springdalelinux  9           amd64  default  20260124_06:38
ubuntu           jammy       amd64  default  20260124_07:42
ubuntu           jammy       arm64  default  20260124_07:42
ubuntu           jammy       armhf  default  20260124_08:02
ubuntu           jammy       riscv64  default  20260124_08:22
ubuntu           noble       amd64  default  20260124_07:42
ubuntu           noble       arm64  default  20260124_07:42
ubuntu           noble       armhf  default  20260124_07:42
ubuntu           noble       riscv64  default  20260124_07:42
ubuntu           plucky      amd64  default  20260124_07:42
ubuntu           plucky      arm64  default  20260124_07:48
ubuntu           plucky      armhf  default  20260124_08:07
ubuntu           plucky      riscv64  default  20260124_08:24
ubuntu           questing    amd64  default  20260124_07:42
ubuntu           questing    arm64  default  20260124_07:42
ubuntu           questing    armhf  default  20260124_07:42
voidlinux        current     amd64  default  20260124_17:10
voidlinux        current     arm64  default  20260124_17:10
```

```
---

Distribution:
debian
Release:
trixie
Architecture:
amd64

Downloading the image index
Downloading the rootfs
Downloading the metadata
The image cache is now ready
Unpacking the rootfs

---
You just created a Debian trixie amd64 (20260124_05:24) container.

To enable SSH, run: apt install openssh-server
No default root or user password are set by LXC.
```

Um den Container beim Booten automatisch starten zu lassen, muss eine Zeile zu seiner Config hinzugefügt werden:

```
echo "lxc.start.auto = 1" >> /var/lib/lxc/<name>/config
```

Damit der Container auch Netzwerk hat, wird ganz normal seine `/etc/systemd/network/eth0.network` angepasst. Z.B:

```
cat >>/var/lib/lxc/<name>/rootfs/etc/systemd/network/eth0.network <<EOF
[Match]
Name=eth0
[Network]
Address=176.9.167.13/28
Gateway=176.9.167.1

DNS=10.10.10.3
DNS=2a01:4f8:262:445e:100::3
Address=2a01:4f8:262:445e:300::d/72
```

```
Gateway=2a01:4f8:262:445e:300::1
EOF
```

Um Probleme beim Neustarten der Container zu vermeiden ist es des Weiteren sinnvoll eine feste MAC-Adresse für das Netzwerk-Interface festzulegen:

```
echo "lxc.net.0.hwaddr = f2:93:8c:e9:93:98" >> /var/lib/lxc/<name>/config
```

Mit `lxc-autostart` kann der Container gestartet werden. Mit `lxc-attach -n jmp100.prima.de` betritt man den Container und installiert alles wichtige, wie einen SSH-Server.

# Übersicht über die Softwarestände der Produktivsysteme

| | | |
|---|---|---|
| buero | Debian GNU/Linux | 13 trixie |
| * blackhole | Debian GNU/Linux | 12 bookworm |
| cogito | Ubuntu | 24.04 Noble |
| laboratory | Debian GNU/Linux | 13 trixie |
| * conf | Ubuntu | 22.04 Jammy |
| * e.ns | Debian GNU/Linux | 12 bookworm |
| * hafen | Debian GNU/Linux | 13 trixie |
| * vserver | Debian GNU/Linux | 13 trixie |
| * zooey | Debian GNU/Linux | 13 trixie |
| * lilly | Debian GNU/Linux | 12 bookworm |
| * lucy | Debian GNU/Linux | 10 buster |
| * mail | Debian GNU/Linux | 13 trixie |

# Hetzner-Server

Auf dem Weg weg von Knipp wurde im August 2022 ein Server bei Hetzner gemietet. Der Server vom Modell AX51 steht im deutschen Rechenzentrum und wir über den Robot-User office@ping.de verwaltet.

Der Server heißt `laboratory.ping.de` (CNAME `lab.ping.de`) und ist über die IP-Adressen `167.235.0.46` und `2a01:4f8:262:445e::2` zu erreicht.

# Technische Daten

- CPU: Octa-Core AMD Ryzen™ 7 3700X
- RAM: 128 GB DDR4 ECC
- HDD: 2 x 12 TB
- SSD: 512 GB NVME SSD
- Netzwerk: 1 Gbit/s
- Primäre IPv4-Adresse und /64-IPv6-Netz

# Software

Das System läuft aktuell mit einem Proxmox 8.14 Image. Dies ist eine auf Debian 12 (Bookworm) basierte Virtualisierungslösung, die eine Web-Oberfläche bietet. Proxmox kann VMs (auf Basis von KVM) und Container (lxc) verwalten. Die Konfiguration richtet sich an der Doku von Proxmox. Verwendete Bridge-Namen und IP-Netze entsprechend (soweit möglich) den Beispielen der Doku.

Das Webinterface von Proxmox kann über https://laboratory.ping.de:8006/ erreicht werden.

Zum Login können die Nutzer des Systems über `pam` genutzt werden. Diese müssen aber wie folgt für Proxmox aktiviert werden:

```
pveum user add <login>@pam
pveum acl modify / --roles PVEAdmin --users <login>@pam
```

# Storage

Bei der Installation des Systems wurde ein RAID-1 über beide HDDs gebildet und in einer LVM-Volumegroup verwendet. Die NVMe-SSD wurde nicht eingebunden und wird später über LVM als Cache für bestimme LVs genutzt (z.B. Mailqueue).

# Netzwerk

Da aktuell nur eine öffentliche IPv4-Adresse verfügbar ist, wird diese an keine VM gebunden, sonder die benötigten Ports werden an die VMs weitergeleitet. Die VMs (und Container) sind an die interne Bridge `vmbr0` gebunden. Hier wird das interne IPv4-Netz `10.10.10.0/24` verwendet. Für IPv6 wird das Netz `2a01:4f8:262:445e:100::0/72` auf die Bridge geroutet.

# haProxy

Für viele VMs/Container ist das Teilen einer IPv4-Adresse relativ problemlos. Da z.B. der Container für DNS andere Ports verwendet als die Mail-VM können diese einfach aufgeteilt werden. Problematischer wird dies mit Port 80 und 443. Diese Ports werden sowohl von der Mail-VM genutzt als auch von einem Webserver-Container. Aus diesem Grund werden Port 80 und 443 nicht an eine VM weitergeleitet, sondern auf dem Host selbst über haProxy als Reverse-Proxy weitergeleitet. Dabei nutzt haProxy für Port 80 den im HTTP-Request angegeben Host und auf 443 wird SNI zur Unterscheidung genutzt. Dies hat den Vorteil, dass haProxy eine Verbindung auf Port 443 an die Ziel-VM weiterleitet, ohne die Verschlüsselung aufzutrennen. Somit bleiben auch sämtliche Daten bis in die VM verschlüsselt und andere VMs können nicht durch Sniffen an der Bridge die Kommunikation einsehen.

Konfiguriert wird haProxy auf dem Basissystem über das Config-File `/etc/haproxy/haproxy.cfg`. Neue VMs oder Namen werden dabei als `backend` konfiguriert. Dies geschieht für Port `80` und `443` getrennt. Für eine Test-VM mit der internen IPv4-Adresse `10.10.10.2` sieht dies wie folgt aus:

```
backend testvm_https
        mode tcp
        option ssl-hello-chk
        source 0.0.0.0 usesrc clientip
        server testvm:443 10.10.10.2:443 weight 100

backend testvm_http
        mode http
```

```
        source 0.0.0.0 usesrc clientip
        server testvm:80 10.10.10.2:80 weight 100
```

Speziell ist dabei das Keyword `usesrc`, wodurch die `tproxy`-Funktion haProxy aktiviert wird. Dies bedeutet, dass die VMs die ursprüngliche IP des Clients sehen und diese auch entsprechend loggen oder filtern können.

Um die Backends zu nutzen, müssen diese in den entsprechenden Frontends für http und https hinzugefügt werden:

```
frontend http
        bind 167.235.0.46:80 transparent
        mode http

        ## exact matches
        use_backend testvm_http if { hdr(Host) -i testvm.ping.de } !{ ssl_fc }

frontend https
        option tcplog
        bind 167.235.0.46:443 transparent
        mode tcp
        tcp-request inspect-delay 5s
        tcp-request content accept if { req_ssl_hello_type 1 }

        ## exact matches
        use_backend testvm_https if { req.ssl_sni -i testvm.ping.de }
```

Über den Teil in den Klammern hinter dem `if` wird der Hostname geprüft.

# Aktuelle VMs/Container

Folgende VMs und Container sind aktuell konfiguriert:

## TestVM

Eine kleine Debian 11 VM mit einem Webserver zum Testen des haProxy. Abgeschaltet wenn nicht benötigt.

## e.ns.ping.de

Container mit konfiguriertem Bind. Dient als Nameserver für `ping.de`, `prima.de`, `ping.ruhr` und `prima.ruhr`, um auch noch dann von Nutzen zu sein, wenn Knipp mal wieder vollständig ausfällt. Die Zonen werden aktuell von den Servern bei Knipp über `AXFR` gezogen sobald diese sich dort ändern.

Im Falle eines Ausfalls von Knipp könnte hier angesetzt werden, um z.B. `www.ping.de` auf einen anderen Webserver umzubiegen.

# mail.ping.ruhr

Debian 11 (Bullseye) VM mit Docker und den Mailcow-Docker-Containern für unseren Test mit den `*.ping.ruhr`- und `*.prima.ruhr`-Domains. Das Mailcow hat alle PING- und Prima-Sites konfiguriert (nur halt mit der `.ruhr`-Endung) und kann entsprechend für Tests genutzt werden.

# lilly

Debian 12 (Bookworm)

# lucy

Debian 10 Buster

News-Server mit architektur-abhängigem Storage 32bit

# hafen

Debian 12.

Docker Container Server:

- portainer,
- Keycloak (auth),
- Nextcloud (cloud),
- mastodon, Derzeit mit Patch beim Starten der etwas längere Nachrichten erlaubt:

```
sed -ie 's/MAX_CHARS = 500/MAX_CHARS = 1500/' app/validators/status_length_validator.rb
```

- Matrix (Synapse Port 8448),
- Jabber (ejabberd),
- bookstack

Dort läuft auch ein nginx, der das SSL terminiert, welches haProxy auf laboratory transparent durchgereicht hat für **auth, code-forgejo, mastodon, matrix, matrix-account** und **wiki** (das liest Du gerade!).

# vserver

VM für Mitglieder VPS

# conf

Big Blue Button BBB Videokonferenz

# zooey

u.a. wiki.ping.de (ehemals techdoc und aktiv wikis)

# Mailserver

Der PING-Mailserver hat seine eigene virtuelle Maschine und nutzt getrennte IP-Adressen, um ein versehentliches Versenden von SPAM durch andere Dienste zu verhindern.

Aktuell hört der Mailserver auf den Namen mail.ping.ruhr und ist eine Debian 13 Trixie VM.

# KI-Server

PING hat 2025 einen KI-Server angeschafft. Er heißt **cogito.ping.de** und befindet sich im Rechnerraum des Gebäudes in der Joseph-von-Fraunhofer-Straße.

# Technische Daten

- CPU AMD Threadripper Pro 5955WX 16 cores 32 threads 4.5Ghz, Zen 3, 64MB L3, 128 PCIe 4.0 lanes
- Mainboard Asus Pro WS WRX80E Sage SE Wifi (7x PCIe 4.0 x16, 8x DDR4 DIMM, 3x M.2, ...)
- 2x RAM Corsair Dominator Platinum RGB White UDIMM 64GB **Kit** DDR4-3600 CL18-19-19-39 (128GB gesamt)
- GPU NVIDIA GeForce RTX 3090 Founders Edition 24 GB
- GPU Zotac Gaming GeForce RTX 3090 Trinity OC 24 GB mit Noctua Lüftern
- Fractal Design Define 7 XL Black TG Dark Tint schallgedämmt Big-Tower (12 PCI Steckplätze)
- SilverStone IceGem 240P AIO CPU-Wasserkühlung
- Antec Neo Eco Gold Modular NE1300G m 1300W ATX 3.0 Netzteil
- 4x SSD Lexar NM790 1TB M.2 NVMe PCIe 4.0 in Asus Hyper M.2 X16 Gen 4 Card (RAID 0)
- SSD Samsung PM951 512GB M.2 NVMe (boot)
- SSD Samsung EVO 850 500GB S-ATA
- 4x Noctua NF-P12 redux-1700 PWM 120mm Lüfter

siehe auch https://geizhals.de/wishlists/3870524

Zu dem Mainboard gehört auch eine PCIe 4.0 x16 Karte um vier PCIe 4.0 x4 NVMe SSDs anzuschließen. Dort befinden sich die 4 Lexar SSDs.

Durch die 2 GPUs stehen derzeit 48GB schnelles VRAM zur Verfügung, eine Erweiterung ist möglich. Ins Gehäuse passen maximal fünf 2-slot GPUs.

Der Hauptspeicher ist auf acht 16GB-Module verteilt und nutzt so die 8 Speicherkanäle der AMD Threadripper Pro Architektur.

Der Platz im Gehäuse und das Mainboard mit vielen PCIe-Lanes ermöglichen es uns, bei Bedarf noch mehr GPUs einzubauen.

# Software

Auf dem Rechner läuft derzeit Ubuntu 24.04 LTS.

Das Konzept sieht vor, die Dienste als Docker Container laufen zu lassen. Die compose Dateien liegen in Unterverzeichnissen unter `/opt`.

Mit `nvtop` kann der Status der GPUs angezeigt werden.

Über das Script `/usr/local/bin/set-gpu-power.sh` sind die GPUs aktuell auf 280W gedrosselt. Das führt zu nur sehr geringen Leistungseinbußen.

# NVidia Kernel Modul

Nach Installation eines neuen Kernels müssem auch die entsprechenden Header installiert werden, damit das Nvidia Kernel Modul kompiliert wird:

```
sudo dpkg-reconfigure nvidia-kernel-common-580
dkms autoinstall
apt install linux-headers-6.17.0-1021-nvidia
modprobe nvidia
```

(Hier am Beispiel von Linux 6.17.0-1021).

Danach ist leider noch ein Reboot vonnöten.

# Storage

Die 4 Lexar NMVe SSDs bilden ein RAID0 das unter `/opt` eingehängt ist. Dadurch können große LLMs schnell in den Speicher geladen werden.

Die S-ATA SSD ist unter `/var/local` eingehängt.

Die PM951 NVMe SSD dient als Bootlaufwerk und enthält auch das OS.

```
$ lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINTS
sda                         8:0    0 465.8G  0 disk
```

```
└─sda1                      8:1    0 465.8G  0 part  /var/local
nvme0n1                   259:0    0 953.9G  0 disk
└─md0                       9:0    0   3.7T  0 raid0 /opt
nvme1n1                   259:1    0 953.9G  0 disk
└─md0                       9:0    0   3.7T  0 raid0 /opt
nvme3n1                   259:2    0 953.9G  0 disk
└─md0                       9:0    0   3.7T  0 raid0 /opt
nvme2n1                   259:3    0 953.9G  0 disk
└─md0                       9:0    0   3.7T  0 raid0 /opt
nvme4n1                   259:8    0 476.9G  0 disk
├─nvme4n1p1               259:9    0     1G  0 part  /boot/efi
├─nvme4n1p2               259:10   0     2G  0 part  /boot
└─nvme4n1p3               259:11   0 473.9G  0 part
  └─ubuntu--vg-ubuntu--lv 252:0    0   445G  0 lvm   /

$ blkid
/dev/nvme0n1: UUID="ebe75b1c-af8f-5e3a-aa0f-9464c3951451" UUID_SUB="1d5de863-0634-7dd3-0e97-7dec
/dev/nvme3n1: UUID="ebe75b1c-af8f-5e3a-aa0f-9464c3951451" UUID_SUB="440d89df-a9d1-bfc9-3634-e57b
/dev/md0: LABEL="RAID" UUID="f57d1a53-8b0c-4119-a02b-e06632c7933d" BLOCK_SIZE="4096" TYPE="ext4"
/dev/nvme2n1: UUID="ebe75b1c-af8f-5e3a-aa0f-9464c3951451" UUID_SUB="eb15293d-d63b-c940-841d-a918
/dev/mapper/ubuntu--vg-ubuntu--lv: UUID="98bd9894-3827-42bb-a0f4-d92931530cab" BLOCK_SIZE="4096"
/dev/nvme1n1: UUID="ebe75b1c-af8f-5e3a-aa0f-9464c3951451" UUID_SUB="797f9137-3a34-2689-5d0b-a294
/dev/sda1: LABEL="ssd500" UUID="c57d324d-3c4f-4f5d-90ca-3859ca87f550" BLOCK_SIZE="4096" TYPE="ex
/dev/nvme4n1p3: UUID="XbVKNc-zwqt-qe2c-fj2e-8MRA-p8e0-XDQdsz" TYPE="LVM2_member" PARTUUID="5f653
/dev/nvme4n1p1: UUID="7006-F657" BLOCK_SIZE="512" TYPE="vfat" PARTUUID="f57a2cbc-da3a-4322-8921-
/dev/nvme4n1p2: UUID="8256bdab-088d-437e-a82b-b94470729f4c" BLOCK_SIZE="4096" TYPE="ext4" PARTUU

$ cat /proc/mdstat
Personalities : [raid0] [linear] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid0 nvme3n1[3] nvme2n1[2] nvme0n1[0] nvme1n1[1]
      4000288768 blocks super 1.2 512k chunks

unused devices: <none>
```

# vLLM mit Open-WebUI

Für Inferenz läuft i.d.R ein vLLM Server. Als WebUI gibt es dafür ein open-webui.

Für das Umwandeln von Office Dokumenten (zum Beispiel ODT) läuft Apache Tika.

Das Docker compose file liegt unter `/opt/vllm/`

Auf milla.ping.de (aka buero) läuft ein nginx der open-webui unter https://ki.ping.de erreichbar macht. Für den Login nutzt bitte unser Single Sign-On.

Der vLLM bietet auch direk auf Port 8000 unter https://ki.ping.de:8000/v1 eine OpenAI-kompatible API. Ihr benötigt das Bearer Token, ihr erhaltet es auf dieser geschützten Wiki-Seite.

# Ollama mit Open-WebUI

Alternativ können wir den Ollama Server starten. Der ist aber nicht mehr der bevorzugte Dienst, u.a. weil die Performance mit mehreren Usern schlecht ist.

Das Docker compose file liegt unter `/opt/ollama/`

Beim Einsatz von Ollama erscheint im Model-Selektor von Open-WebUI ein grüner Punkt neben den LLMs, die derzeit im GPU Speicher sind.

Das Script `/usr/local/bin/ollama-nogpu.sh` ist dafür da, den Ollama Container neu zu starten, falls dieser mal wieder die GPUs nicht erkennt.

# ComfyUI

ComfyUI (primär für KI-Bildergenerierung) ist noch nicht fertig installiert, es liegt unter `/opt/comfyui` und kann bei Bedarf gestartet werden. Vorher sollte ollama gestoppt werden, weil nicht genügend GPU VRAM für beide Dienste gleichzeitig vorhanden ist.