Serverexperimente II

Dokumentiert den Aufbau meines öffentlichen Linux-Severs, gehostet bei 1blu. Weiterhin werden meine Serverexperimente in der Hetzner Cloud und mit meinen beiden lokalen Raspis beschrieben.

- Kubuntu mit Parallels auf Apple Silicon
- Ubuntu/Xubuntu WLAN für iMac
- Ubuntu 24.04 Server
    - Bestellung
    - Grundlegende Absicherung
    - Docker installieren
    - Traefik Crowdsec Stack einrichten
- Raspi 3
    - Raspi OS Trixi lite: Waveshare 3,5'' Display im Terminal Mode

Kubuntu mit Parallels auf Apple Silicon

Thought i would post this since i've been trying to figure it out.

1. Get a copy of ubuntu server arm64
2. Install the "normal" option
3. sudo apt update
4. sudo apt upgrade
5. install parallels tools
    1. click the alarm looking button
    2. it will put in the CD
    3. sudo mkdir /media/cdrom
    4. sudo mount -o exec /dev/cdrom /media/cdrom
    5. cd /media/cdrom
    6. sudo ./install
    7. reboot
6. sudo apt update
7. sudo apt upgrade
8. sudo apt install kubuntu-desktop
9. reboot (shutdown -r now)
10. Internet works, but Discover update system has problems
11. Open /etc/netplan/50-installer-config.yaml
12. add "renderer: NetworkManager" after the "version: 2" and delete “ethernets: “ section with the adapter name and dhcp setting. (Network manager will add it automatically). Snap will not work if u don’t delete ethernets.
13. to avoid a two minute boot wait penalty, disable the servers default systemd networkd service (sudo systemctl disable systemd-networkd.service), which will be replaced by the NetworkManager service on next boot.
14. save and reboot
15. had to re-install parallels tools manually. Mount the iso located here and run the install gui script: Applications > Parallels Desktop > Contents > Resources > Tools > pr-tools-lin-arm.iso
16. remove fwupd and fwupd-discover backend with `sudo apt-get remove fwupd plasma-discover-backend-fwupd`. Without that the "get updates" function of discover will hang.
17. reboot
18. thats it

Ubuntu/Xubuntu WLAN für iMac

Der iMac ist von 2014 und funktioniert unter Linux noch sehr gut. Allerdings muss für den WLAN-Chip ein separater Treiber geladen werden, da die Kernel den von Apple verbauten WLAN-Chip von Broadcom nicht unterstützen. Mittlerweile ist der passende Treiber im Archiv gelandet. Mit folgenden Befehlen kann er (noch) geladen und aktiviert werden:

```
wget https://archive.ubuntu.com/ubuntu/pool/restricted/b/broadcom-sta/broadcom-sta-dkms_6.30.223.271-23ubuntu1.2_all.deb
sudo dpkg -i broadcom-sta-dkms_6.30.223.271-23ubuntu1.2_all.deb
```

Seit August 2026 gibt es eine neuere Version des Treibers (...ubuntu1.3_all.deb). Es bietet sich also an, vor dem Download im Archiv nachzusehen, welche Version die aktuellste ist.

Wenn man wissen möchte, welcher WLAN-Chip im System steckt, hilft der Befehl ***sudo inxi -Fxxxz*** weiter.

Ubuntu 24.04 Server

Beschreibt, wie man bei Hetzner einen virtuellen Server mit Ubuntu 24.04 LTS anlegt, diesen grundlegend absichert und einen mit Crowdsec abgesicherten Docker Traefik Stack einrichtet.

Bestellung

Über die Hetzner Cloud Console kann ein neuer Server geordert oder aus einem Snapshot erstellt werden. In der Regel werden aus Kostengründen die neuen ARM-Server bevorzugt, aber noch ist die Softwarewelt nicht so weit, ARM CPUs in allen Bereichen zu unterstützen. Für den Fall der Fälle kann auf AMD- oder Intel-Hardware ausgewichen werden.

Ein üblicher Standard-Server sieht etwa wie folgt aus:

Standort	Nürnberg, Falkenstein oder Helsinki
Image	Ubuntu 22.04
Typ	Shared CPU Arm64 (Ampere) CAX11 (2 Cores, 4 GB RAM, 40 GB SSD, 20 TB Traffic)
Networking	IP4 optional / IP6 optional / Private Network optional (je nach Bedarf)
Platzierungsgruppen	Empfehlenswert, wenn mehrere Server im Projekt laufen (z. B. Kubernetes)
SSH-Keys	Konfigurierte Public Keys von allen administrierenden Systemen (empfohlen)
Firewalls	Je nach Einsatz - für öffentliche Server ein Muss!
Backups	täglich, 7 Tage Zyklus. Empfehlenswert - Kosten: 20% vom Serverpreis
Name	nach eigenem Ermessen
Kosten	5,29€ / Monat mit IP4-Adresse und Backups aktiviert (Stand April 2024)

Nach Abschluss einer Bestellung wird der Server innerhalb weniger Sekunden erstellt und seine Basisdaten sowie seine IP-Adresse(n) angezeigt. Mit der IP-Adresse kann sich mittels ssh auf ihm angemeldet werden:

```
ssh root@<IP-Adresse>
```

Eine IPv6 Adresse wird nicht ganz korrekt angezeigt. Am Ende erscheint ein ::/64, was bedeuten würde das das letzte Paket 0000 lautet. Dem ist nicht so, denn das letzte Paket des Adressblocks lautet korrekt 0001.
Um sich die sperrige IP-Adresse nicht merken zu müssen, empfiehlt es sich, einen passenden Namen mit der IP-Adresse in Verbindung zu bringen und in der Shell zu exportieren. Für einen dauerhaften Export benötigt es nur einen Eintrag in der Datei .zshrc (oder .bashrc):

```
export myserver=<IP-Adresse>
```

Damit ist dann der ssh Verbindungsaufbau mit dem Befehl ssh root@$myserver möglich.

Der Login erfordert dank des übergebenen öffentlichen SSH-Key kein Passwort, es sein denn, der ssh-Key selbst ist geschützt. Von Hetzner wird bei Angabe eines ssh-Keys kein Password generiert und per E-Mail verschickt.

Im nächsten Schritt wird der Server grundlegend abgesichert.

Grundlegende Absicherung

Externe Server bzw. generell alle Server, die am Internet hängen, sollten Anmeldungen auf der Konsole ausschließlich über einen SSH-Schlüssel zulassen. Zudem sollte sich der User root gar nicht von außen anmelden können. Außerdem sollte eine Firewall Zugriffe nur über explizit erlaubte Ports erlauben und eine Software installiert sein, die Angriffsversuche erkennt und blockiert. Am Beispiel eines Hetzner Cloud vServers unter Ubuntu 22.04 LTS wird hier beschrieben, wie ein solcher Server abgesichert werden kann.

Anlegen eines neuen Users mit root Rechten

Erfreulicherweise können bei Hetzner, wie auf der Seite Hetzner vServer bestellen & absichern beschrieben, in der Bestellung eines Servers bereits alle SSH-Schlüssel, die auf dem Server erlaubt sein sollen, mit angegeben werden. Voraussetzung dafür ist, dass die öffentlichen Schlüssel im Hetzner-Account abgelegt wurden. Diese Option sollte unbedingt genutzt werden, denn so ist der Server gleich von Beginn an mit einem Grundschutz gegen Attacken von außen versehen, denn eine Anmeldung als root ohne einen passenden privaten Schlüssel ist nicht mehr möglich.

Nachdem der Server das erstmals gestartet wurde, erfolgt die Anmeldung mit `ssh root@<ip_adresse>`. Dazu ist es unbedingt erforderlich, dass ein gültiger öffentlicher Schlüssel im .ssh Verzeichnis des aufrufenden Users liegt. Dafür hat Hetzner bei der Anlage des Servers gesorgt. Alle in der Bestellung angegebenen öffentlichen Schlüssel sind in der Datei ~/.ssh/authorized_keys eingetragen.

Nach einer erfolgreichen Anmeldung wird als User root auf dem Server gearbeitet. Aus diesem Grund enthalten die nachfolgend aufgeführten Kommandos auch kein vorangestelltes sudo.

Als erste Aktion nach einer Neuinstallation und der Erstanmeldung als root ist unbedingt das Betriebssystem zu aktualisieren. Dazu ist der Befehl `apt update && apt upgrade -y` auszuführen. Unter Umständen weist das Upgrade am Ende darauf hin, dass ein Reboot des Systems erforderlich ist. Das kann schnell mit dem Befehl `reboot` herbeigeführt werden.

Im nächsten Schritt ist die Firewall zu konfigurieren und aktivieren. Die Firewall ufw ist bereits installiert, aber noch nicht aktiviert. Bevor sie aktiviert werden kann, sind allerdings noch ein paar Konfigurationen erforderlich. Eingehender Verkehr ist nur auf den explizit zugelassenen Ports zulässig, ausgehender Verkehr wird generell erlaubt.

```
# Eingehenden Verkehr generell verbieten
sudo ufw default deny incoming
```

```
# Ausgehenden Verkehr generell eerlauben
sudo ufw default allow outgoing

# SSH Verbindungen zulassen
sudo ufw allow ssh

# Wenn auch HTTP/HTTPS erlaubt sein soll
sudo ufw allow http
sudo ufw allow https

# Firewall aktivieren
sudo ufw enable
```

Im nächsten Schritt ist ein User mit sudo Rechten einzurichten:

```
#
# Benutzer 'sysadmin' einrichten und zum sudoer machen.
# Dabei wird vorausgesetzt, dass es sich beim eingelogten
# user root und dem neuen user sysadmin um ein und dieselbe
# Person handelt.
#
adduser sysadmin
usermod -a -G sudo sysadmin
mkdir /home/sysadmin/.ssh
cp ~/.ssh/authorized_keys /home/sysadmin/.ssh/
chmod 700 /home/sysadmin/.ssh
chmod 400 /home/sysadmin/.ssh/authorized_keys
chown sysadmin:sysadmin /home/sysadmin/.ssh
chown sysadmin:sysadmin /home/sysadmin/.ssh/authorized_keys
```

In einem neuen Terminal sollte nun ein ssh Anmeldeversuch mit dem neuen User erfolgen. Die Anmeldung muss ohne Eingabe eines Passworts gelingen, es sei denn, das System fragt nach dem Passwort für den ssh Key, wenn dieser mit einem Passwort abgesichert ist.

Nach erfolgreicher Anmeldung muss abschließend geprüft werden, ob der neue User sysadmin mittels sudo mit root-Rechten arbeiten kann. Um dem User zukünftig die Passworteingabe nach einem sudo Befehl zu ersparen, führen folgende Schritte zum Ziel:

```
#
# Eine Datei im Verzeichnis /etc/sudoers.d anlegen
#
```

```
sudo vi /etc/sudoers.d/sysadmin-user

#
# In dieser Datei folgende Zeile einfügen:
#
sysadmin ALL=(ALL) NOPASSWD:ALL

#
# In einem neuen Terminalfenster testen:
#
sudo -i
```

> Der Verzicht auf das sudo Password ist eine Schwächung der Systemsicherheit!

> Die Nachfolgenden Schritte dürfen nur durchgeführt werden, wenn sich der neue User per ssh ohne Password mit seinem ssh Key anmelden und root Rechte einnehmen kann. Ansonsten droht der Ausschluss aus dem Server!.

Anmeldung des Users root verbieten

Ein nächster wichtiger Schritt hin zu einem sicheren System ist das Unterbinden des Login als User root und des Login mit Passwort. Ein Login soll nur noch mittels ssh Schlüssel möglich sein. Dazu sind in der Datei */etc/ssh/sshd_config*folgende Einträge wie dargestellt zu ändern:

```
#
# Anmelden mit root verbieten und nur publickey erlauben
#
PermitRootLogin no
AuthenticationMethods publickey
PasswordAuthentication no
```

Nach der Änderung muss der ssh Service mit dem Befehl `sudo systemctl restart sshd` neu gestartet werden.

Docker installieren

Die Services, die von Servern bereitgestellt werden, sind meist in Form von Docker Images implementiert. Hierzu gehören z. B. Blogs, Wissensdatenbanken, Git-Repositories und viele mögliche mehr. Voraussetzung ist eine aktuelle Docker Engine, die idealerweise von Docker selbst bezogen wird, denn Erfahrungsgemäß sind die in den Ubuntu Repositorien vorhandenen Paket nicht aktuell genug. Dazu werden die Docker Repositorien in die Ubuntu-Repositorien eingebunden, so dass mit jedem apt update/upgrade die aktuellste Version installiert wird.

Das nachfolgende Skript (*install_docker.sh*) erledigt die Aufnahme des Docker Repositories sowie die Installation der Docker Engine automatisch. Allerdings wird erwartet, das die Installation noch einmal bestätigt wird.

```
#/bin/bash!
#-----------------------------------------------
# install_docker.sh
#
# Install docker using the apt repository
# Brings all commands of the docker side together
# in one script.
#-----------------------------------------------

# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

```
# Install the latest version
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin
```

Es bietet sich an, das Skript auf der Docker-Seite ab und an auf Aktualität zu überprüfen.

Zur Ausführung des **docker** Kommandos werden root Rechte benötigt. Soll das Kommando ohne vorangestelltes **sudo** direkt aufgerufen werde können, ist die Gruppe Docker einzurichten und den ausgewählten Benutzern zuzuweisen.

```
# Create the docker group
sudo groupadd docker

# Add user to the docker group
sudo usermod -aG docker $USER
```

> Bitte beachten, dass Benutzende mit der Zuweisung der docker Gruppe root Rechte erlangen!

Ubuntu 24.04 Server

Traefik Crowdsec Stack einrichten

Der traefik-crowdsec-stack soll einen sicheren Betrieb von Docker Services ermöglichen. Dabei wird als Reverse-Proxy und Load-Balancer die derzeit aktuellste Version 3.x.x von Traefik eingesetzt. Die Kommunikation der von Traefik gerouteten Services wird mit Crowdsec in eigenen Netzwerken abgesichert. Schließlich wird der Stack um die von Außen erreichbaren Dienste "Ralfs Blogs", "Bines Blog", "Bücherregal" und "Gitea-Server" erweitert. Alle hierzu benötigten Docker Services können bequem über eine einzelne Docker Compose Datei gesteuert werden.

Die Basis für die hier verwendeten Texte und den Code bildet der Artikel Traefik V3 Installation, Konfiguration und CrowdSec-Security von psycho0verload. Es sei nicht verschwiegen das einige Elemente des Artikels kostenpflichtig sind. Hierzu gehört insbesondere der Netzwerk-Abschnitt.

Einrichten der grundlegenden Stack-Struktur

Bevor es richtig losgeht noch ein wichtiger Hinweis: Alle Shell-Befehle sind mit root Privilegien auszuführen. In den Skripten wird kein vorangestelltes sudo verwendet. Das Skript selbst ist mit sudo aufzurufen.

Um den Stack anzulegen, wird das Hauptverzeichnis traefik-crowdsec-stack unter /opt/containers/ erstellt und in dieses Verzeichnis gewechselt. Danach werden die erforderlichen Unterverzeichnisse und Dateien erzeugt, um den Stack mit crowdsec, socket-proxy und traefik zu konfigurieren. Schließlich werden noch die korrekten Berechtigungen für sensible Zertifikatsdateien gesetzt.

Anschließend werden die in der Struktur angelegten Konfigurations- und Environmentdateien mit Inhalten gefüllt. Dabei werden Parameter wie Passwörter oder API-Schlüssel mit einem in doppelt geschweiften Klammern versehenem Kommentar versehen. Diese Parameter sind im Nachgang von Hand anzupassen.

Um den Stack vorzubereiten, kann das nachfolgende Skript ausgeführt werden (zum Aufklappen auf den Namen klicken):

create_stack.sh

```
#!/bin/bash
#
# ensure that user has root privileges
#
if [ $(id -u) -ne 0 ]
  then echo Please run this script as root or using sudo!
  exit
fi

#
# create traefic-crowdsec-stack directory structure
#
mkdir -p /opt/containers/traefik-crowdsec-stack && cd /opt/containers/traefik-crowdsec-
stack
mkdir -p compose data/{crowdsec/{config,data},socket-
proxy,traefik/{certs,dynamic_conf},traefik-crowdsec-bouncer,service-
blog/{bine,ralf},service-bookstack/{app,db},service-gitea}

#
# Create some files and change permissions of cert related files
#
touch .env docker-compose.yml \
compose/{crowdsec.yml,networks.yml,socket-proxy.yml,traefik-crowdsec-
bouncer.yml,traefik.yml} \
data/{crowdsec/.env,socket-
proxy/.env,traefik/{.env,.htpasswd,traefik.yml,certs/{acme_letsencrypt.json,tls_letsencryp
t.json},dynamic_conf/{http.middlewares.default.yml,http.middlewares.default-security-
headers.yml,http.middlewares.gzip.yml,http.middlewares.traefik-
bouncer.yml,http.middlewares.traefik-dashboard-auth.yml,tls.yml}},traefik-crowdsec-
bouncer/.env}

chmod 600 data/traefik/certs/{acme_letsencrypt.json,tls_letsencrypt.json}

#
# Fill master dotenv file
#
cat << 'EOF' > /opt/containers/traefik-crowdsec-stack/.env
ABSOLUTE_PATH=${PWD}
TZ=Europe/Berlin
```

```
SERVICES_TRAEFIK_LABELS_TRAEFIK_HOST=HOST(`traefik.rastef.de`)
EOF

#
# Fill master compose file
#
echo "include:
  - compose/service-blog-bine.yml
  - compose/service-blog-ralf.yml
  - compose/service-bookstack.yml
  - compose/service-gitea.yml
  - compose/crowdsec.yml
  - compose/socket-proxy.yml
  - compose/traefik-crowdsec-bouncer.yml
  - compose/traefik.yml
  - compose/networks.yml" >> /opt/containers/traefik-crowdsec-stack/docker-compose.yml


#
# Fill Crowdsec compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/crowdsec.yml
services:
  crowdsec:
    container_name: ${SERVICES_CROWDSEC_CONTAINER_NAME:-crowdsec}
    env_file: ${ABSOLUTE_PATH}/data/crowdsec/.env
    hostname: ${SERVICES_CROWDSEC_HOSTNAME:-crowdsec}
    healthcheck:
      test: ["CMD", "cscli", "version"]
      timeout: 2s
      interval: 20s
      retries: 5
      start_period: 10s
    image: ${SERVICES_CROWDSEC_IMAGE:-
crowdsecurity/crowdsec}:${SERVICES_CROWDSEC_IMAGE_VERSION:-latest}
    networks:
      crowdsec:
        ipv4_address: ${SERVICES_CROWDSEC_NETWORKS_CROWDSEC_IPV4:-172.31.127.254}
        ipv6_address: ${SERVICES_CROWDSEC_NETWORKS_CROWDSEC_IPV6:-
```

```
fd00:1:be:a:7001:0:3e:6fff}
      socket_proxy:
        ipv4_address: ${SERVICES_TRAEFIK_NETWORKS_SOCKET_PROXY_IPV4:-172.31.255.252}
        ipv6_address: ${SERVICES_TRAEFIK_NETWORKS_SOCKET_PROXY_IPV6:-
fd00:1:be:a:7001:0:3e:8ffd}
    restart: unless-stopped
    security_opt:
      - no-new-privileges=true
    volumes:
      - ${ABSOLUTE_PATH}/data/crowdsec/config:/etc/crowdsec
      - ${ABSOLUTE_PATH}/data/crowdsec/data:/var/lib/crowdsec/data
      - /var/log/auth.log:/var/log/auth.log:ro
      - /var/log/traefik:/var/log/traefik:ro
EOF

#
# Fill Socket-Proxy compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/socket-proxy.yml
services:
  socket-proxy:
    container_name: ${SERVICES_SOCKET_PROXY_CONTAINER_NAME:-socket-proxy}
    env_file: ${ABSOLUTE_PATH}/data/socket-proxy/.env
    hostname: ${SERVICES_SOCKET_PROXY_HOSTNAME:-socket-proxy}
    healthcheck:
      test: ["CMD-SHELL", "wget -q --spider http://localhost:2375/_ping || exit 1"]
      timeout: 3s
      interval: 20s
      retries: 3
      start_period: 10s
    image: ${SERVICES_SOCKET_PROXY_IMAGE:-lscr.io/linuxserver/socket-
proxy}:${SERVICES_SOCKET_PROXY_IMAGE_VERSION:-latest}
    networks:
      socket_proxy:
        ipv4_address: ${SERVICES_SOCKET_PROXY_NETWORKS_SOCKET_PROXY_IPV4:-172.31.255.254}
        ipv6_address: ${SERVICES_SOCKET_PROXY_NETWORKS_SOCKET_PROXY_IPV6:-
fd00:1:be:a:7001:0:3e:8fff}
    read_only: true
    restart: unless-stopped
```

```
    tmpfs:
      - /run
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
EOF

#
# Fill Traefik Crowdsec Bouncer compose file
#
cat << 'EOF' > /opt/containers/traefik-crowdsec-stack/compose/traefik-crowdsec-bouncer.yml
services:
  traefik_crowdsec_bouncer:
    container_name: ${SERVICES_TRAEFIK_CROWDSEC_BOUNCER_CONTAINER_NAME:-
traefik_crowdsec_bouncer}
    depends_on:
      crowdsec:
        condition: service_healthy
    env_file: ${ABSOLUTE_PATH}/data/traefik-crowdsec-bouncer/.env
    hostname: ${SERVICES_TRAEFIK_CROWDSEC_BOUNCER_HOSTNAME:-traefik-crowdsec-bouncer}
    image: ${SERVICES_TRAEFIK_CROWDSEC_BOUNCER_IMAGE:-ghcr.io/freifunkmuc/traefik-
crowdsec-bouncer}:${SERVICES_TRAEFIK_CROWDSEC_BOUNCER_IMAGE_VERSION:-latest}
    networks:
      crowdsec:
        ipv4_address: ${SERVICES_TRAEFIK_CROWDSEC_BOUNCER_NETWORKS_CROWDSEC_IPV4:-
172.31.127.252}
        ipv6_address: ${SERVICES_TRAEFIK_CROWDSEC_BOUNCER_NETWORKS_CROWDSEC_IPV6:-
fd00:1:be:a:7001:0:3e:6ffd}
    restart: unless-stopped
EOF

#
# Fill Traefik compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/traefik.yml
services:
  traefik:
    container_name: ${SERVICES_TRAEFIK_CONTAINER_NAME:-traefik}
    depends_on:
      crowdsec:
```

```
        condition: service_healthy
      socket-proxy:
        condition: service_healthy
    env_file: ${ABSOLUTE_PATH}/data/traefik/.env
    hostname: ${SERVICES_TRAEFIK_HOSTNAME:-traefik}
    healthcheck:
      test: ["CMD", "traefik", "healthcheck", "--ping"]
      timeout: 1s
      interval: 10s
      retries: 3
      start_period: 10s
    image: ${SERVICES_TRAEFIK_IMAGE:-traefik}:${SERVICES_TRAEFIK_IMAGE_VERSION:-latest}
    labels:
      traefik.enable: "true"
      traefik.http.routers.traefik-dashboad.entrypoints: websecure
      traefik.http.routers.traefik-dashboad.middlewares: traefik-dashboard-auth@file
      traefik.http.routers.traefik-dashboad.rule: ${SERVICES_TRAEFIK_LABELS_TRAEFIK_HOST:-
HOST(`traefik.yourdomain.com`)}
      traefik.http.routers.traefik-dashboad.service: api@internal
      traefik.http.routers.traefik-dashboad.tls: "true"
      traefik.http.routers.traefik-dashboad.tls.certresolver: tls_resolver
      traefik.http.services.traefik-dashboad.loadbalancer.sticky.cookie.httpOnly: "true"
      traefik.http.services.traefik-dashboad.loadbalancer.sticky.cookie.secure: "true"
      traefik.http.routers.pingweb.rule: PathPrefix(`/ping`)
      traefik.http.routers.pingweb.service: ping@internal
      traefik.http.routers.pingweb.entrypoints: websecure
    networks:
      crowdsec:
        ipv4_address: ${SERVICES_TRAEFIK_NETWORKS_CROWDSEC_IPV4:-172.31.127.253}
        ipv6_address: ${SERVICES_TRAEFIK_NETWORKS_CROWDSEC_IPV6:-
fd00:1:be:a:7001:0:3e:6ffe}
      proxy:
        ipv4_address: ${SERVICES_TRAEFIK_NETWORKS_PROXY_IPV4:-172.31.191.254}
        ipv6_address: ${SERVICES_TRAEFIK_NETWORKS_PROXY_IPV6:-fd00:1:be:a:7001:0:3e:7fff}
      socket_proxy:
        ipv4_address: ${SERVICES_TRAEFIK_NETWORKS_SOCKET_PROXY_IPV4:-172.31.255.253}
        ipv6_address: ${SERVICES_TRAEFIK_NETWORKS_SOCKET_PROXY_IPV6:-
fd00:1:be:a:7001:0:3e:8ffe}
    ports:
```

```
      - mode: host
        target: 80
        published: "80"
        protocol: tcp
      - mode: host
        target: 443
        published: "443"
        protocol: tcp
    restart: unless-stopped
    security_opt:
      - no-new-privileges:true
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - ${ABSOLUTE_PATH}/data/traefik/traefik.yml:/etc/traefik/traefik.yml
      - ${ABSOLUTE_PATH}/data/traefik/.htpasswd:/etc/traefik/.htpasswd
      -
${ABSOLUTE_PATH}/data/traefik/certs/acme_letsencrypt.json:/etc/traefik//acme_letsencrypt.j
son
      -
${ABSOLUTE_PATH}/data/traefik/certs/tls_letsencrypt.json:/etc/traefik/tls_letsencrypt.json
      - ${ABSOLUTE_PATH}/data/traefik/dynamic_conf:/etc/traefik/dynamic_conf:ro
      - /var/log/traefik/:/var/log/traefik/
EOF

#
# Fill Crowdsec compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/networks.yml
networks:
  crowdsec:
    name: ${NETWORKS_CROWDSEC_NAME:-crowdsec}
    driver: bridge
    enable_ipv6: true
    ipam:
      driver: default
      config:
      - subnet: ${NETWORKS_CROWDSEC_SUBNET_IPV4:-172.31.64.0/18}
      - subnet: ${NETWORKS_CROWDSEC_SUBNET_IPV6:-fd00:1:be:a:7001:0:3e:6000/116}
    attachable: true
```

```
  proxy:
    name: ${NETWORKS_PROXY_NAME:-proxy}
    driver: bridge
    enable_ipv6: true
    ipam:
      driver: default
      config:
      - subnet: ${NETWORKS_PROXY_SUBNET_IPV4:-172.31.128.0/18}
      - subnet: ${NETWORKS_PROXY_SUBNET_IPV6:-fd00:1:be:a:7001:0:3e:7000/116}
    attachable: true
  socket_proxy:
    name: ${NETWORKS_SOCKET_PROXY_NAME:-socket_proxy}
    driver: bridge
    enable_ipv6: true
    ipam:
      driver: default
      config:
      - subnet: ${NETWORKS_SOCKET_PROXY_SUBNET_IPV4:-172.31.192.0/18}
      - subnet: ${NETWORKS_SOCKET_PROXY_SUBNET_IPV6:-fd00:1:be:a:7001:0:3e:8000/116}
    attachable: true
    internal: true
EOF

#
# Fill Service Blog Bine compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/service-blog-bine.yml
services:
  bine:
    container_name: ${SERVICES_BINE_CONTAINER_NAME:-service-blog-bine}
    image: nginx:latest
    labels:
      traefik.docker.network: proxy
      traefik.enable: true
      traefik.http.routers.bine-secure.entrypoints: websecure
      traefik.http.routers.bine-secure.rule: Host(`bine.rastef.de`)
      traefik.http.routers.bine-secure.service: service-blog-bine
      traefik.http.routers.bine-secure.tls: true
      traefik.http.routers.bine-secure.tls.certresolver: tls_resolver
```

```
      traefik.http.routers.bine.entrypoints: web
      traefik.http.routers.bine.rule: Host(`bine.rastef.de`)
      traefik.http.services.bine.loadbalancer.server.port: "80"
    networks:
      default: null
      proxy: null
    restart: unless-stopped
    volumes:
      - ${ABSOLUTE_PATH}/data/service-blog/bine:/usr/share/nginx/html:ro
EOF

#
# Fill Service Blog Ralf compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/service-blog-ralf.yml
services:
  blog:
    container_name: ${SERVICES_BLOG_CONTAINER_NAME:-service-blog-ralf}
    image: nginx:latest
    labels:
      traefik.docker.network: proxy
      traefik.enable: true
      traefik.http.routers.blog-secure.entrypoints: websecure
      traefik.http.routers.blog-secure.rule: Host(`rastef.de`) || Host(`www.rastef.de`)
      traefik.http.routers.blog-secure.service: service-blog-ralf
      traefik.http.routers.blog-secure.tls: true
      traefik.http.routers.blog-secure.tls.certresolver: tls_resolver
      traefik.http.routers.blog.entrypoints: web
      traefik.http.routers.blog.rule: Host(`rastef.de`) || Host(`www.rastef.de`)
      traefik.http.services.blog.loadbalancer.server.port: "80"
    networks:
      default: null
      proxy: null
    restart: unless-stopped
    volumes:
      - ${ABSOLUTE_PATH}/data/service-blog/ralf:/usr/share/nginx/html:ro
EOF

#
```

```
# Fill Service Bookstack compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/service-bookstack.yml
services:
  books:
    container_name: bookstack
    image: linuxserver/bookstack:latest
    environment:
      - TZ=Europe/Berlin
      - APP_URL=https://buecher.rastef.de
      - APP_KEY={{get it from Bookstack with: docker run -it --rm --entrypoint /bin/bash
lscr.io/linuxserver/bookstack:latest appkey}}
      - DB_HOST=bookstackdb
      - DB_DATABASE=bookstack
      - DB_USERNAME=bookstack
      - DB_PASSWORD={{--Das Bookstack DB Password--}}
    labels:
      traefik.docker.network: proxy
      traefik.enable: true
      traefik.http.routers.books-secure.entrypoints: websecure
      traefik.http.routers.books-secure.rule: Host(`buecher.rastef.de`)
      traefik.http.routers.books-secure.service: bookstack
      traefik.http.routers.books-secure.tls: true
      traefik.http.routers.books-secure.tls.certresolver: tls_resolver
      traefik.http.routers.books.entrypoints: web
      traefik.http.routers.books.rule: Host(`buecher.rastef.de`)
      traefik.http.services.books.loadbalancer.server.port: "80"
    networks:
      default: null
      proxy: null
    restart: unless-stopped
    volumes:
      - ${ABSOLUTE_PATH}/data/bookstack/app:/config
  bookstackdb:
    container_name: booksdb
    image: linuxserver/mariadb:latest
    environment:
      - TZ=Europe/Berlin
      - MYSQL_ROOT_PASSWORD={{--Das MySQL Root Password--}}
```

```
      - MYSQL_DATABASE=bookstack
      - MYSQL_USER=bookstack
      - MYSQL_PASSWORD={{--Das Bookstack DB Password--}}
    networks:
      default: null
    restart: unless-stopped
    volumes:
      - ${ABSOLUTE_PATH}/data/bookstack/db:/config
EOF

#
# Fill Service Gitea compose file
#
cat <<'EOF' > /opt/containers/traefik-crowdsec-stack/compose/service-gitea.yml
services:
  gitea:
    container_name: ${SERVICES_GITEA_CONTAINER_NAME:-service-gitea}
    image: docker.gitea.com/gitea:latest-rootless
    labels:
      traefik.docker.network: proxy
      traefik.enable: true
      traefik.http.routers.gitea-secure.entrypoints: websecure
      traefik.http.routers.gitea-secure.rule: Host(`gitea.rastef.de`)
      traefik.http.routers.gitea-secure.service: service-gitea
      traefik.http.routers.gitea-secure.tls: true
      traefik.http.routers.gitea-secure.tls.certresolver: tls_resolver
      traefik.http.routers.gitea.entrypoints: web
      traefik.http.routers.gitea.rule: Host(`gitea.rastef.de`)
      traefik.http.services.gitea.loadbalancer.server.port: "3000"
    networks:
      default: null
      proxy: null
    restart: unless-stopped
    volumes:
      - ${ABSOLUTE_PATH}/data/gitea/repos:/var/lib/gitea
      - ${ABSOLUTE_PATH}/data/gitea/config:/etc/gitea
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
EOF
```

Mit dem ***tree*** Kommando lässt sich die Verzeichnisstruktur verifizieren. Wenn das ***tree*** Kommando fehlt, kann es schnell mit `sudo apt install tree` installiert werden.

```
$ tree -L 4 -a /opt/containers/traefik-crowdsec-stack/
```

```
/opt/containers/traefik-crowdsec-stack/
??? compose
?   ??? crowdsec.yml
?   ??? networks.yml
?   ??? service-blog-bine.yml
?   ??? service-blog-ralf.yml
?   ??? service-bookstack.yml
?   ??? service-gitea.yml
?   ??? socket-proxy.yml
?   ??? traefik-crowdsec-bouncer.yml
?   ??? traefik.yml
??? data
?   ??? crowdsec
?   ?   ??? config
?   ?   ??? data
?   ?   ??? .env
?   ??? service-blog
?   ?   ??? bine
?   ?   ??? ralf
?   ??? service-bookstack
?   ?   ??? app
?   ?   ??? db
?   ??? service-gitea
?   ??? socket-proxy
?   ?   ??? .env
?   ??? traefik
?   ?   ??? certs
?   ?   ?   ??? acme_letsencrypt.json
?   ?   ?   ??? tls_letsencrypt.json
?   ?   ??? dynamic_conf
?   ?   ?   ??? http.middlewares.default-security-headers.yml
?   ?   ?   ??? http.middlewares.default.yml
?   ?   ?   ??? http.middlewares.gzip.yml
?   ?   ?   ??? http.middlewares.traefik-bouncer.yml
?   ?   ?   ??? http.middlewares.traefik-dashboard-auth.yml
?   ?   ?   ??? tls.yml
?   ?   ??? .env
?   ?   ??? .htpasswd
?   ?   ??? traefik.yml
?   ??? traefik-crowdsec-bouncer
?       ??? .env
??? docker-compose.yml
??? .env

18 directories, 25 files
```

> Bitte beachten, dass in diversen Dateien noch manuelle Korrekturen vorgenommen werden müssen. Daher bitte noch kein docker compose ausführen. Lesen sie dazu auf der nächsten Seite weiter.

Raspi 3

Raspi OS Trixi lite: Waveshare 3,5'' Display im Terminal Mode

Ausgangslage

An einem Raspberry PI 3B (R3) soll das "3.5 inch Resistive Touch Display (B) V2" mit einer Auflösung von 480×320 Pixeln angeschlossen werden. Das IPS Display für die SPI-Schnittstelle soll an diesem Rechner nur die Standard-Console anzeigen, da der Raspi mit seinen mageren 1GB Speicher mit einem grafischen Desktop überfordert ist.

Wird die Möglichkeit, das Display auch als Terminal zu benutzen, schon nur versteckt beschrieben, ist dummerweise ein dafür benötigtes Framebuffer-Tool nicht mehr kompilierbar. Selbst eine mit allen Patches versehene Betriebssystemversion, auf dem die Beispiele beruhen, kann das Tool nicht mehr starten.

Installation

Basis ist die 64 Bit Lite Version von Debian Trixi. Das Linux Betriebssystem wird mit dem Raspberry PI Imager auf eine Micro SD Speicherkarte (oder ein USB-Device wie z. B. eine externe SSD) geschrieben. Beim Erstellen des Image sollte darauf geachtet werden, dass der Username dem Default ***pi*** entspricht.

Der R3 wird mit der Karte (oder dem USB-Device) gebootet und anschließend mit dem üblichen Zweiklang auf den neuesten Stand gebracht. Und um sicher zu gehen, dass auch alle Aktualisierungen geladen sind, sollte der R3 neu gestartet werden.

```
sudo apt-get update && sudo apt-get upgrade -y
sudo rebtot
```

Nun ist der R3 bereit für die Installation der für das Display benötigten Software. Zuerst wird der eigentliche Display-Treiber vom Wavershare Fileserver geholt und in das Raspi-OS Treiberverzeichnis kopiert. Da es sich bei der Datei um eine zip-Datei handelt, muss sie vor dem

kopieren entpackt werden. Beide Aktionen erledigt ***unzip*** elegant in einem Befehl:

```
wget https://files.waveshare.com/upload/1/1e/Waveshare35b-v2.zip
sudo unzip ./Waveshare35b-v2.zip -d /boot/overlays
```

Der Treiber muss noch richtig eingebunden und das HDMI-Interface auch dann aktiviert werden, wenn kein HDMI-Kabel angeschlossen ist. Dazu ist die Firmware Konfiguration zu bearbeiten:

```
sudo nano /boot/firmware/config.txt
```

In dieser Datei sind folgende zwei Aktionen durchzuführen:

```
#---------------------------------------
# First step: Comment out following two lines
#---------------------------------------
# dtoverlay=vc4-kms-v3d
# max_framebuffers=2

#---------------------------------------
# Second step: Add following lines to the end
#---------------------------------------
dtparam=spi=on
dtoverlay=waveshare35b-v2
hdmi_force_hotplug=1
max_usb_current=1
hdmi_group=2
hdmi_mode=1
hdmi_mode=87
hdmi_cvt 480 320 60 6 0 0 0
hdmi_drive=2
display_rotate=0
```

Durch die geänderte Konfiguration ist nun sichergestellt, dass beim Systemstart die Devices für die Framebuffer (/dev/fb0 und /dev/fb1) angelegt werden. Um der Standard-Konsole (tty1) nun den passenden Framebuffer (/dev/fb1) zuzuweisen, kann das Tool ***con2fbmap*** genutzt werden. Da diese Zuordnung bei jedem Bootvorgang automatisch vorgenommen werden soll, erstellen wir einen neuen Service:

```
sudo nano /etc/systemd/system/console-display.service
```

Diese Datei füllen wir mit folgendem Inhalt:

```
[Unit]
Description=Framebuffer /dev/fb1 für Console tty1 aktivieren
After=network.target

[Service]
Type=simple
User=pi
ExecStart=/usr/bin/con2fbmap 1 1
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Den neu erstellten Service müssen wir noch aktivieren, damit er bei jedem Boot ausgeführt wird:

```
sudo systemctl enable console-display.service
```

Jetzt nur noch das Auto-Login setzen und dann den R3 neu starten:

```
sudo raspi-config nonint do_boot_behaviour B2 # auto-login
sudo reboot
```

Nach einer gewissen Zeit - ein R3 ist nicht der schnellste Rechner - sollte das Display die für ein Linux-System üblichen Konsole-Ausgaben anzeigen.

Quellen:

Display Dokumentation:	https://www.waveshare.com/wiki/3.5inch_RPi_LCD_(B)
Erklär-Video	https://www.youtube.com/watch?v=qU6_zLMUzLA
Erklär-Video als Text (Kurzform)	https://github.com/TemporarilyOffline/WaveShare
Kompilierungsproblem mit con2fbmap Lösung	https://github.com/goodtft/LCD-show/issues/404
con2fbmap Beschreibung	https://manpages.debian.org/trixie/fbset/con2fbmap.1.en.html
Service	https://raspberry.tips/raspberrypi-einsteiger/raspberry-pi-autostart-von-skripten-und-programmen-einrichten
Berrybase Gehäuse für Display	https://www.berrybase.de/gehaeuse-fuer-raspberry-pi-3-3b-und-3-5-display-schwarz

Berrybase Display	https://www.berrybase.de/3-5-display-fuer-raspberry-pi-mit-resistivem-touchscreen