Schritt-für-Schritt-Anleitungen

- PatchMon - Linux Patch Monitoring made Simple
- ESP32-Jarolift (Rollladen-Server)
- QNAP-NAS als Backup-Storage in Proxmox-Backup-Server (PBS) einbinden
- GHome Steckdose mit Tasmota flashen
- Prometheus installieren, Mailcow-Exporter installieren und die "Mailcow" in Grafana visuell darstellen
- MailPiler-Installation

PatchMon - Linux Patch Monitoring made Simple

Um PatchMon zu installieren ist eigentlich nicht viel Arbeit von Nöten.
Einfach einen VPS bestellen (Port 3000 darf nicht in Verwendung sein, interne IPs gehen nicht - es muss eine öffentlich IPv4 sein!).

Als Erstes müssen wir die Maschine updaten:
`apt update -y`
`apt upgrade -y`
`apt install curl -y`

Außerdem brauchen wir "Docker Compose":
`curl -fsSL https://get.docker.com -o get-docker.sh`
`sh get-docker.sh`

Anschließend eine docker-compose.yml erstellen:

docker-compose.yml

Diese dann pullen und ausrollen:
`docker compose up -d`

Das wars schon.
Nun ist PatchMon unter der Adresse `http://***IPv4***:3000` erreichbar.

Updates:
`docker compose up -d --pull`

ESP32-Jarolift (Rollladen-Server)

Inbetriebnahme:

1. Nutze für die Stromversorgung ein USB-C Kabel und ein beliebiges 5 V – Netzteil. **Mein Tipp:** schließe den Controller an deinen WLAN-Router. Stromversorgung und super Empfang.
2. Der ESP erstellt automatisch einen eigenen Accesspoint namens "ESP32-Jarolift".
3. Verbinde dich mit einem Smartphone oder Computer mit diesem Netz und gib im Browser die IP-Adresse des ESPs ein (**192.168.4.1**). Jetzt hast du die Möglichkeit, unter **Einstellungen -> WiFi** dein WLAN-Netz mit dem entsprechenden Passwort einzugeben. Starte anschließend den ESP neu.
4. Der ESP bekommt von dem Router eine neue IP-Adresse, die man in den WLAN-Einstellungen des Routers findet und ist darunter erreichbar. **Achtung:** aus mir unbekannten Gründen, heißt der „ESP32-Jarolift" im Router esp32-xxxxxx (Zahlenfolge).
5. Nachdem du die Weboberfläche über die neue IP-Adresse geöffnet hast, muss du unter **Einstellungen -> Jarolift** den MasterKey eingeben:
   MasterMSB - 27193a9b
   MasterLSB - 117c0835
6. Die Seriennummer ist unwichtig, ich nutze 00000a.
7. Setze den Rollladenmotor in den Lernmodus (Fernbedienung gleichzeitig hoch und runter + 8 x Stop) und drücke dann im Reiter **Rollladen** auf den **anlernen** Button.
8. Fertig! Viel Erfolg!

QNAP-NAS als Backup-Storage in Proxmox-Backup-Server (PBS) einbinden

Um ein QNAP NAS als Backup-Storage in Proxmox Backup Server (PBS) hinzuzufügen, musst du den QNAP NAS als Remote-Storage einbinden, indem du entweder NFS oder CIFS/SMB als Netzwerkprotokoll verwendest.
Wir benutzen hier den Zugang über CIFS/SMB.

Schritte zum Hinzufügen eines QNAP NAS als Storage in Proxmox Backup Server:

1. Zugang zum NAS sicherstellen
   Stelle sicher, dass dein QNAP NAS korrekt eingerichtet ist und du über die notwendigen Berechtigungen verfügst, um auf die Freigaben zuzugreifen. Du kannst das über das QNAP Web-Interface konfigurieren.

2. Freigabe auf dem QNAP NAS erstellen
   Erstelle eine Freigabe auf deinem QNAP NAS (wenn noch nicht geschehen):
   Für CIFS/SMB: Stelle sicher, dass die Windows-Freigabe aktiviert ist.
   Gehe zu Control Panel > Network & File Services > Win/Mac/NFS und aktiviere Microsoft Networking (SMB).
   Erstelle eine SMB/CIFS-Freigabe für den Ordner, den du als Backup-Speicher verwenden möchtest.

Proxmox Backup Server einrichten

Zuerst müssen wir einen Mount-Punkt im PBS anlegen, in den der QNAP dann gemountet wird. Das machen wir mit:
`mkdir /mnt/qnap`

Anschließend mounten wir den QNAP in diesen Mount-Pfad:
`mount.cifs -o user=***USERNAME***,pass=***PASSWORT*** //192.168.1.100/***FREIGABEORDNER*** /mnt/qnap`

Wenn das geklappt hat, müssen wir noch eine Datei mit dem Benutzernamen und Passwort anlegen, damit sich der PBS diese Informationen beim automatischen Mounten holen kann:
`nano /etc/credentials-qnap`

Der Inhalt darf nur so aussehen (drauf achten, dass keine Leerzeichen etc. vorhanden sind!):
`username=***USERNAME***`

`password=***PASSWORT***`

Nun müssen wir noch die Rechte der anpassen, damit diese nicht verändert werden können:
`chmod 0600 /etc/credentials-qnap`

Jetzt fügen wir in der `/etc/fstab` noch den Mountbefehl hinzu, damit das Verzeichnis permanent gemountet wird, vor allem nach einem Neustart.
Dazu tragen wir folgendes in die `/etc/fstab` ein:
`//192.168.1.100/***FREIGABEORDNER*** /mnt/qnap cifs seal,cache=none,iocharset=utf8,rw,credentials=/etc/credentials-qnap,uid=0,gid=0,file_mode=0777,dir_mode=0777,vers=3.0,noperm, 0 0`

Storage in der GUI hinzufügen

1. Gehe zu "Datastorer" > "Datastore hinzufügen".
2. Vergib einen Namen (z.B.: `QNAP`)
3. Der Backing-Pfad ist der Mount-Pfad (also hier: `/mnt/qnap`)
4. Den GC-Zeitplan habe ich auf `daily` gelassen
5. Bei Prune-Zeitplan habe ich mich ebenfalls für `daily` entschieden
6. Unter dem Reiter "Prune-Optionen" möchte ich die Backups der letzten 4 Tage behalten, also habe ich bei "Tage behalten" `4` eingetragen
7. Zurück unter "Allgemein" kann man jetzt noch beim Kommentar eintragen, dass dieser Storage eben zum QNAP geht, ist aber kein Muss
8. Nun auf "Hinzufügen" klicken

Das kann jetzt einige Zeit dauern, da der PBS jetzt die "Chunks" im Ordner anlegt.
Anschließend sollte der Storage im PBS hinzugefügt worden sein.

Zur Sicherheit, damit die Berechtigungen vom PBS korrekt an den QNAP übertragen werden und wir schreiben und lesen dürfen, habe ich den PBS einmal rebootet.

Fertig.

Nun können wir endlich Backups über den PBS auf dem QNAP anlegen.

Hinweis:

Bei mir hat der PBS noch etwas rumgezickt und mir eine Fehlermeldung bei der Datastore-Übersicht rausgeschmissen, die da hieß:
`unable to open chunk store at "/mnt/qnap/.chunks" - No such file or directory (os error 2)`

oder:
`Permission denied (os error 13)`

Das Problem habe ich mit einem `systemd Mount-Service` gelöst:
`nano /etc/systemd/system/mount-qnap.service`

Dort folgendes eintragen und speichern:

```
[Unit]
Description=Mount QNAP SMB Share
After=network.target
Requires=network.target

[Service]
Type=oneshot
ExecStart=/bin/mount /mnt/qnap
RemainAfterExit=true

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

Dann noch zwei Befehle zum "Enablen" und "Starten" des Mount-Services:

```
systemctl enable mount-qnap.service
systemctl start mount-qnap.service
```

Mit `systemctl status mount-qnap.service` können wir überprüfen, ob der Service gestartet ist.
Jetzt sollte auch der QNAP korrekt im PBS angezeigt werden.

Das wars.

GHome Steckdose mit Tasmota flashen

Schritt-für-Schritt-Anleitung

Flashen der freien Firmware Tasmota auf smarte Steckdosen von GHome EP2-A.

> “
>
> **Achtung!**
>
> Das Verändern der Firmware eines Geräts kann immer dazu führen, dass das Gerät unbrauchbar wird. Im schlimmsten Fall kann es danach gar nicht mehr benutzt werden! Die Modifikation geschieht auf eigene Gefahr, wir übernehmen keinerlei Haftung für Schäden!

Was ist Tasmota?

Tasmota ist eine freie Firmware, die auf Geräte installiert (geflasht) werden kann, die auf dem Chip ESP8266 basieren. Günstige smarte Steckdosen beispielsweise hängen häufig von einer nicht ganz so komfortablen China-App ab und können ggf. auch nur über das Internet gesteuert werden. Vorteile von Tasmota:

- Keine Abhängigkeit vom Internet.
- Kann vollständig im lokalen Netzwerk betrieben und gesteuert werden.
- Bietet eine einfache Weboberfläche, die im Browser aufgerufen werden kann.
- Lässt sich einfach in Smart Home-Systeme wie Home Assistant einbinden.

Dieser Beitrag zeigt den Installationsprozess von Tasmota am Beispiel der bei Amazon erhältlichen smarten Steckdosen GHome EP2-A (Kein Affiliate-Link).

Voraussetzungen

Folgendes sollte vorhanden sein, um die Steckdosen erfolgreich mit Tasmota flashen zu können:

- Ein WLAN-fähiges Gerät. (hier ein Raspberry Pi 4)
- Ein Smartphone (hier ein iPhone) oder beliebiges anderes zusätzliches WLAN-fähiges Gerät.
- Die Steckdosen, die geflasht werden sollen.

Wenn zusätzlich auch die Strommessfunktion genutzt werden soll, muss diese kalibriert werden, dazu benötigt:

- Ein Strommessgerät, von dem Daten wie Watt, Volt und Ampere abgelesen werden können.
- Einen Verbraucher, der möglichst konstante Strommengen zieht, z.B. eine 60W Lampe.

Wie man den Raspberry Pi installiert und sich damit über SSH verbindet, könnt ihr in unzähligen Anleitung via Google-Suche nachlesen.

Benötigte Software

Zunächst muss ein wenig Software auf dem Raspberry Pi (oder einem beliebigen anderen Debian-basierten System) installiert werden. Dazu verbinden wir uns per SSH auf das Gerät. Da das Flash-Tool viele Abhängigkeiten hat, ist die Durchführung mit Docker am einfachsten. Sollte Docker noch nicht installiert sein, nutzt den nachfolgenden Schnelldurchlauf:

```
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Zusätzlich muss auf jeden Fall `git` auf dem System sein:

```
sudo apt install git
```

Jetzt kann das eigentlich Tool **tuya-convert** von Github geladen werden. Dies kann einfach im Home-Verzeichnis gemacht werden:

```
git clone https://github.com/ct-Open-Source/tuya-convert
```

Anschließend wechseln wir in das neue Verzeichnis:

```
cd tuya-convert
```

Anpassung an tuya-convert

Jedes Gerät in einem Netzwerk hat eine MAC-Adresse. An den ersten Stellen einer MAC-Adresse (Präfix) kann man ableiten, von welchem Hersteller das Gerät kommt. Das Tool **tuya-convert** hat eine Liste dieser Präfixe hinterlegt, um flashbare Geräte identifizieren zu können. Die Steckdosen von GHome scheinen aber MAC-Adressen zu haben, dessen Präfixe dort noch nicht enthalten sind. Deswegen erkennt das Tool diese nicht und wir müssen die Präfixe ergänzen.

In den Amazon Rezensionen ist die Rede davon, dass der Präfix `34987A` hinzugefügt werden muss. Bei meinen Steckdosen war es der Präfix `3CE90E`. Deswegen fügen wir beide in der Liste hinzu, es kann aber sein dass ihr dort noch einen anderen Präfix eintragen müsst, dies erfahrt ihr dann aber erst später im Prozess.

```
nano scripts/oui/esp.txt
```

Und dort ganz am Ende jeweils in eine eigene Zeile die beiden zusätzlichen Präfixe eintragen. Mit **STRG + O -> Eingabetaste -> STRG + X** *(Mac: command statt STRG)* speichern und den Editor beenden.

Steckdose flashen

Jetzt ist nahezu alles vorbereitet, um den Flash-Vorgang zu starten.

.env

Um das Flash-Tool mit Docker zu starten, wird noch eine .env Datei benötigt. In unserem aktuellen Verzeichnus **tuya-convert** befindet sich dazu eine Beispieldatei mit dem Namen `.env-template`, diese kopieren wir:

```
cp .env-template .env
```

Der wichtigste Eintrag in dieser Datei ist der Name des WLAN-Adapters auf unserem System. Voreingestellt ist dort `wlan0` und das sollte auf dem Raspberry Pi und vielen anderen Systemen auch genau so passen. Alle Adapter anzeigen lassen kann man sich unter anderem mit `ip addr`. Sollte der Name abweichen, müsst ihr den vorab in der neuen Datei `.env` anpassen.

Docker Image bauen und Container starten

Jetzt ist endlich alles bereit für den Start. Das Docker Image wird direkt auf dem System gebaut, dann ein Container erstellt und dieser gestartet. Sobald der Vorgang abgeschlossen ist wird der Container wieder automatisch gelöscht:

Hinweis
Der Build-Vorgang kann beim ersten Mal sehr lange dauern. Auf meinen Raspberry Pi 4 lag die Dauer bei 4-5 Minuten.

```
sudo docker compose build && sudo docker compose run --rm tuya
```

Es kann sein, dass noch einige Warnhinweise bestätigt werden müssen. Sobald die nachfolgende Meldung erscheint, müssen zunächst die dort erwähnten Schritte durchgeführt werden, die nachfolgend auch detaillierter beschrieben werden, bevor es mit “Enter” weitergehen kann.

```
IMPORTANT
1. Connect any other device (a smartphone or something) to the WIFI vtrust-flash
   This step is IMPORTANT otherwise the smartconfig may not work!
2. Put your IoT device in autoconfig/smartconfig/pairing mode (LED will blink fast). This is usually done by
pressing and holding the primary button of the device
   Make sure nothing else is plugged into your IoT device while attempting to flash.
3. Press ENTER to continue
```

1. Das Tool hat jetzt ein WLAN mit dem Namen **vtrust-flash** geöffnet. Mit diesem verbinden wir uns jetzt mit dem Smartphone oder einem anderen beliebigen WLAN-fähigen Gerät.
2. Dann stecken wir die smarte Steckdose ist eine stromführende Steckdose. Sie sollte blau blinken und befindet sich damit im Pairing-Modus. Falls nicht, den Button an der Steckdose lange drücken, bis die smarte Steckdose einmal neu startet.

3. Ist beides erledigt, kann die Meldung auf dem Raspberry Pi mit der Eingabetaste bestätigt werden.
4. Jetzt läuft der vollautomatische Flashing-Prozess, bei dem vorerst nur ein minimales Zwischensystem aufgespielt wird.

Sobald das abgeschlossen ist, erscheint folgende Meldung auf dem Raspberry Pi:

```
You can also provide your own image by placing it in the /files directory
Please ensure the firmware fits the device and includes the bootloader
MAXIMUM SIZE IS 512KB

Available options:
  0) return to stock
  1) flash espurna.bin
  2) flash tasmota.bin
  q) quit; do nothing
Please select 0-2: 2
Are you sure you want to flash tasmota.bin? This is the point of no return [y/N] y
```

Da wir Tasmota auf dem Gerät haben möchten, wählen wir Option 2 und bestätigen dies anschließend nochmal mit “y”. Jetzt wird Tasmota auf das Gerät übertragen, anschließend startet die Steckdose neu. Die Frage, ob ein weiteres Gerät geflasht werden soll, kann mit “n” beantwortet werden.

Vorgang bricht ab und findet kein Gerät

Wenn es zu keinem Problem gekommen ist, kann dieser Abschnitt ignoriert werden.

Möglicherweise erhaltet ihr folgende Meldung, bevor der Flash-Vorgang starten kann:

```
Timed out while waiting for the device to (re)connect
==========================================================
Attempting to diagnose the issue...
No ESP82xx based devices connected according to your wifi log.
Here is a list of all the MAC addresses that connected:
3c:e9:0e:f2:23:b5
a6:02:6b:4b:b6:38
```

Damit muss in der Datei `scripts/oui/esp.txt` ggf. ein weiterer MAC-Präfix eingetragen werden. Die Meldung gibt aus, welche beiden Geräte gefunden worden sind, hier:

- 3c:e9:0e:f2:23:b5
- a6:02:6b:4b:b6:38

Eins davon wird die Steckdose sein, das andere das Smartphone. Dort lässt sich leicht nachschauen, welche MAC-Adresse das Gerät hat, bei meinem iPhone war es die `a6:02:6b:4b:b6:38`, damit gehört die andere Adresse zur Steckdose. Der Präfix sind die ersten drei Blöcke ohne Doppelpunkt: `3c:e9:0e:f2:23:b5` = `3CE90E`

Nach der Ergänzung kann der Vorgang dann einfach wiederholt werden, ggf. muss die Steckdose wieder durch langes Drücken des Buttons in den Pairing-Modus versetzt werden.

Konfiguration

Die Steckdose läuft jetzt mit Tasmota. Jetzt folgen noch ein paar Schritte, um sie korrekt zu konfigurieren.

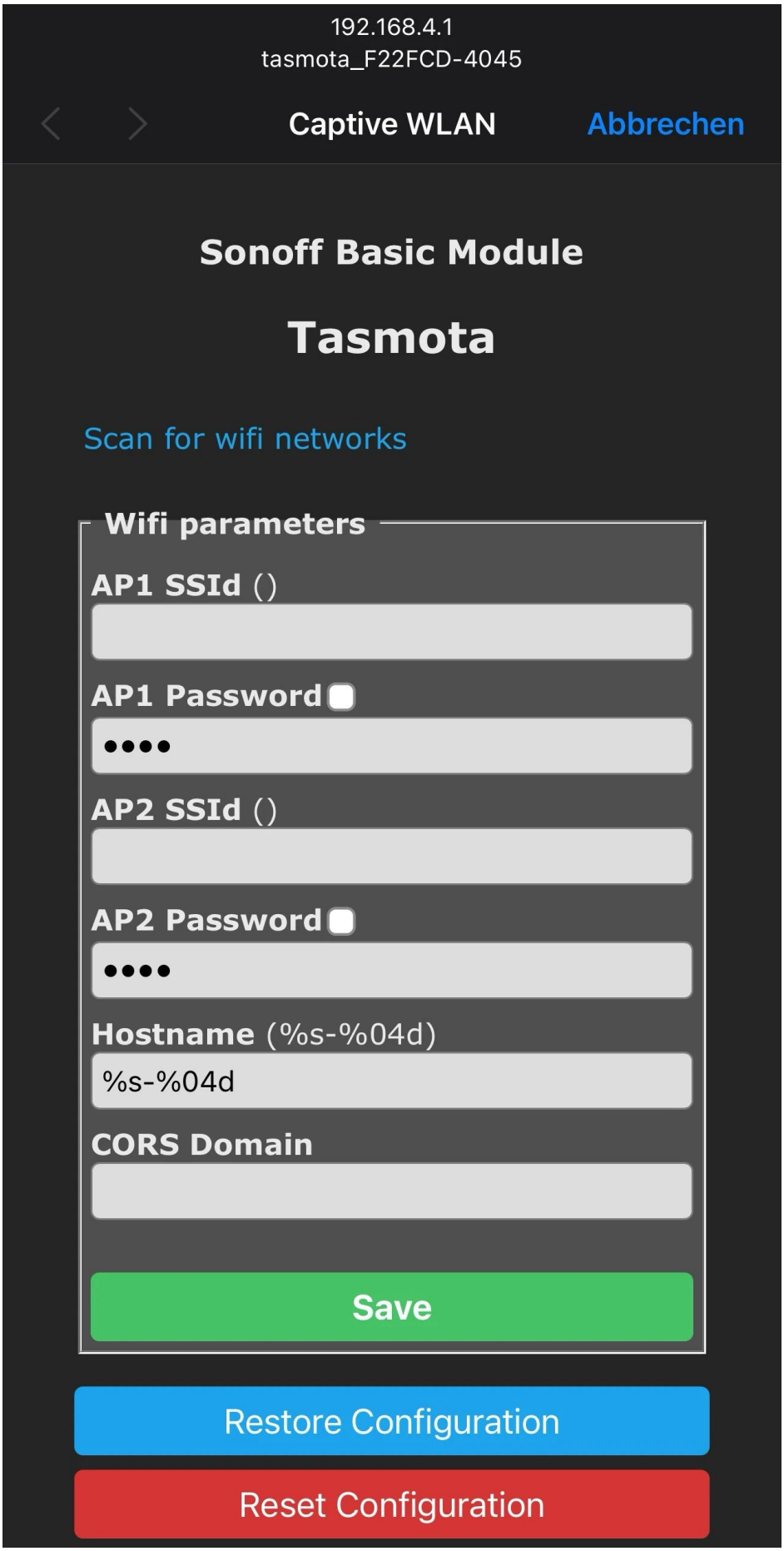
Steckdose mit WLAN verbinden

Um die Steckdose mit dem gewünschten WLAN zu verbinden, sind folgende Schritte notwendig:

1. Die neu geflashte Steckdose öffnet ein WLAN mit einem Namen, der mit “tasmota_xxxxx-xxxx” startet.

2. Das Smartphone mit diesem WLAN der Steckdose verbinden.

3. Es öffnet sich nach dem Verbinden automatisch eine Webseite zum Eintragen der Login-Daten für das gewünschte WLAN, mit das sich die Steckdose verbinden soll (kann etwas dauern). Falls nicht, folgende IP im Browser aufrufen: http://192.168.4.1
4. **Scan for wifi networks** auswählen -> Es wirkt so als wenn nichts passiert. Einfach etwas länger warten, er sucht nach WLANs und zeigt diese dann an.
5. WLAN auswählen, dann das dazugehörige Passwort bei **AP1** eingeben.
6. Mit Save bestätigen. Die Steckdose startet neu, danach ist die Steckdose im WLAN und per Browser erreichbar.
7. Die IP der Steckdose kann im Router gefunden werden.


192.168.4.1
tasmota_F22FCD-4045
Captive WLAN
Abbrechen
Sonoff Basic Module
Tasmota
Scan for wifi networks
Wifi parameters
AP1 SSId ()
AP1 Password
AP2 SSId ()
AP2 Password
Hostname (%s-%04d)
%s-%04d
CORS Domain
Save
Restore Configuration
Reset Configuration

Firmware updaten

Die geflashte Version von Tasmota ist sehr alt, deswegen ist der erste Schritt jetzt, die Firmware zu aktualisieren. Dazu öffnen wir in einem Browser die Weboberfläche der Steckdose, indem wir dessen IP eingeben. Ggf. kommt eine Warnung weil es sich um eine unverschlüsselte HTTP-Verbindung handelt, dann muss einmal bestätigt werden dass die Seite wirklich aufgerufen werden soll.

Auf der Oberfläche dann **Firmware Upgrade** auswählen und dann auf den oberen Button bei der OTA-URL auf **Start upgrade** drücken. Jetzt gibt es zwei Möglichkeiten:

1. Entweder startet jetzt das vollautomatische Update und die Steckdose startet anschließend neu.
2. Oder es wird eine Datei **tasmota.bin.gz** im Browser heruntergeladen. Dann muss im unteren Feld diese Datei ausgewählt werden und dort **Start upgrade** gedrückt werden, damit dann der Updateprozess startet und die Steckdose neu startet.

Template anpassen

Nach dem Update kommt man wieder auf die Weboberfläche. Damit Tasmota weiß, wie alle Funktionen korrekt angesteuert werden, muss das Template angepasst werden.

Dazu auf **Configuration** -> **Configure other** und im ersten Feld “Template” folgendes eintragen *(gilt nur für die GHome-Steckdosen, bei anderen Geräten ggf. abweichend)*:

```
{"NAME":"GHome EP2-A","GPIO":[576,0,320,0,2656,2720,0,0,2624,32,0,224,0,0],"FLAG":0,"BASE":45}
```

Direkt da drunter bei **Activate** muss ein Häkchen gesetzt werden. Zusätzlich können **Device Name** und **Friendly Name 1** angepasst werden. Die Steckdose an der mein PC ist hat zum Beispiel den Namen “tasmota-pc”.

Dann wird das ganze mit **Save** bestätigt, das Gerät startet dann wieder neu.

Strommessung kalibrieren (optional)

Wenn auch die Strommessung sinnvolle Werte ergeben soll, muss dies einmal kalibriert werden. Wie schon erwähnt wird dazu ein Spannungsmessgerät und ein konstanter Verbraucher benötigt, wie zum Beispiel eine 60W Lampe.

Vorgehen dabei:

1. Tasmota Steckdose in eine Steckdose stecken, darin dann das Spannungsmessgerät einstecken und darin dann wiederum den Verbraucher (Lampe o.ä.).
2. In der Weboberfläche den Menüpunkt **Console** aufrufen.

In der Konsole folgende Befehle eintippen (mit Eingabetaste jeweils bestätigen):

- `savedata 1`
- `VoltageSet ###.##` (in V)
- `PowerSet ###.##` (in W)
- `CurrentSet ###.##` (in mA)
- `savedata 0`

Anschließend wieder ins Hauptmenü gehen, dort sollten jetzt passende Stromwerte zu sehen sein.

Abschluss

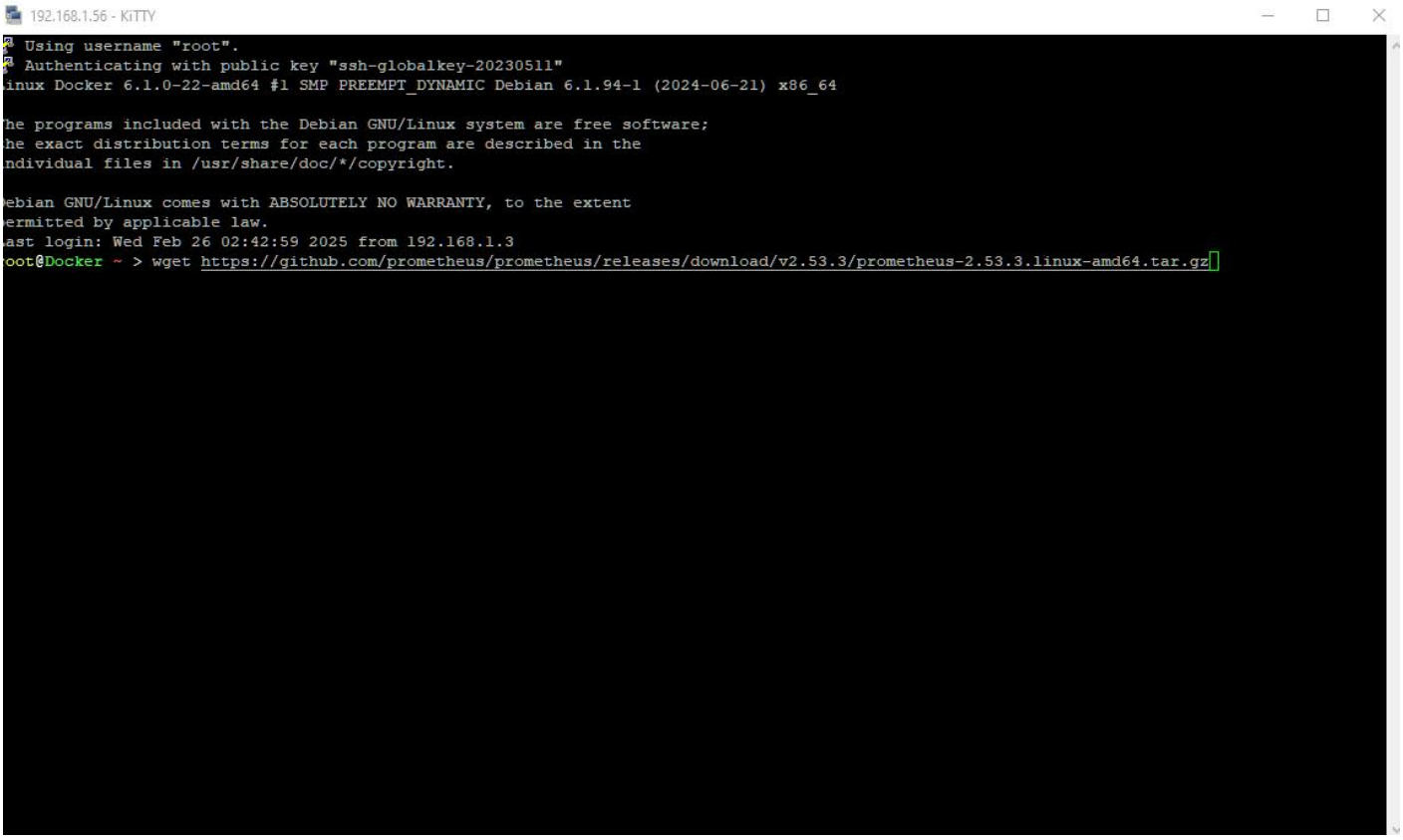
Die smarte Steckdose läuft jetzt mit Tasmota und ist einsatzbereit. Sie kann jetzt problemlos über die Weboberfläche gesteuert werden, eine Einbindung in ein Smart Home-System ist ebenfalls kein Problem mehr.

Prometheus installieren, Mailcow-Exporter installieren und die "Mailcow" in Grafana visuell darstellen

Also meine Grundlage ist eine VM auf Proxmox mit Docker und Docker-Compose. Wobei Letzteres nicht benötigt wird.
Ebenfalls empfehle ich immer ein Snapshot / Backup anzulegen! :)

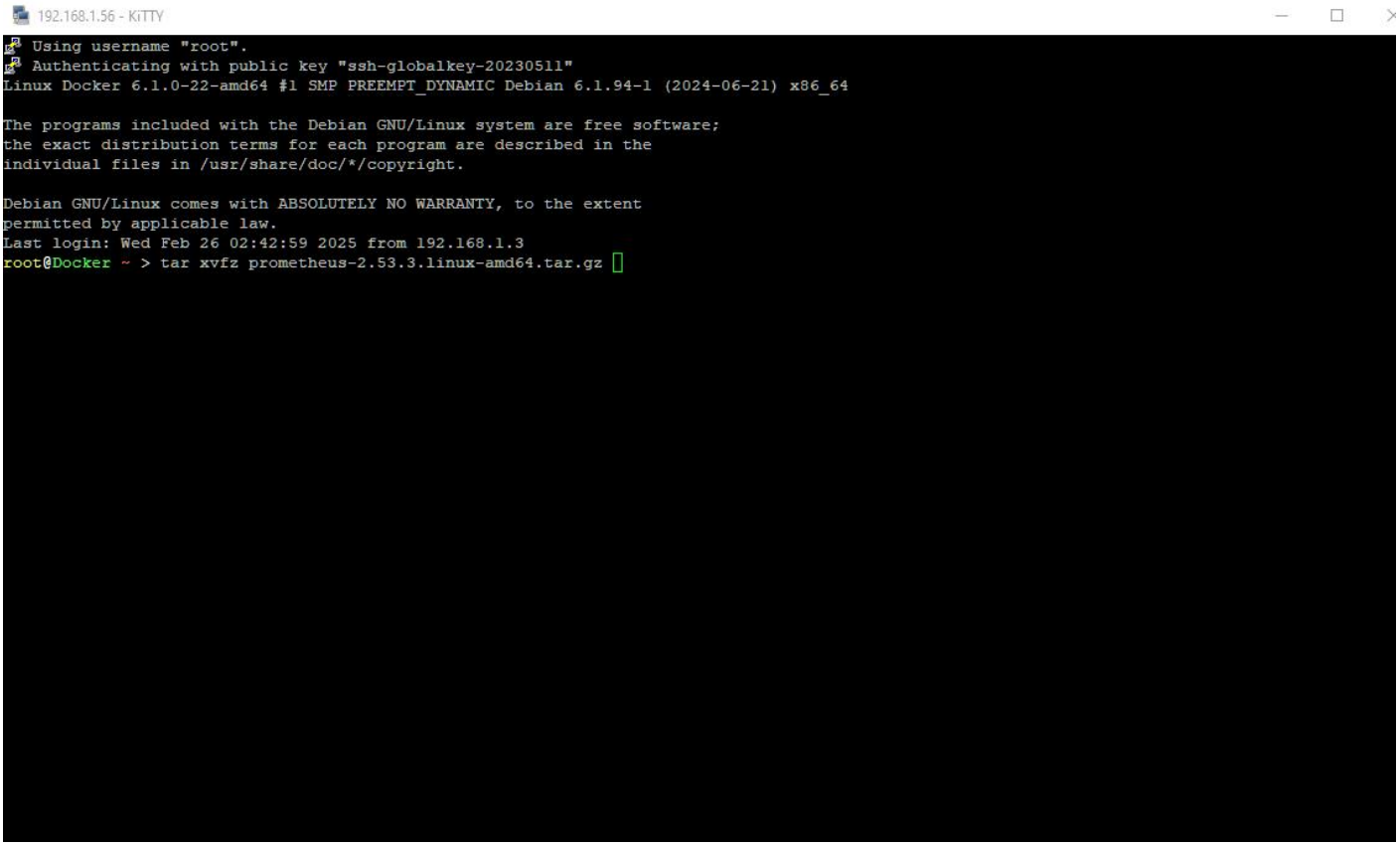
Auf der Docker-VM habe ich erstmal Prometheus installiert:

```
wget https://github.com/prometheus/prometheus/releases/download/v2.53.3/prometheus-2.53.3.linux-amd64.tar.gz
```



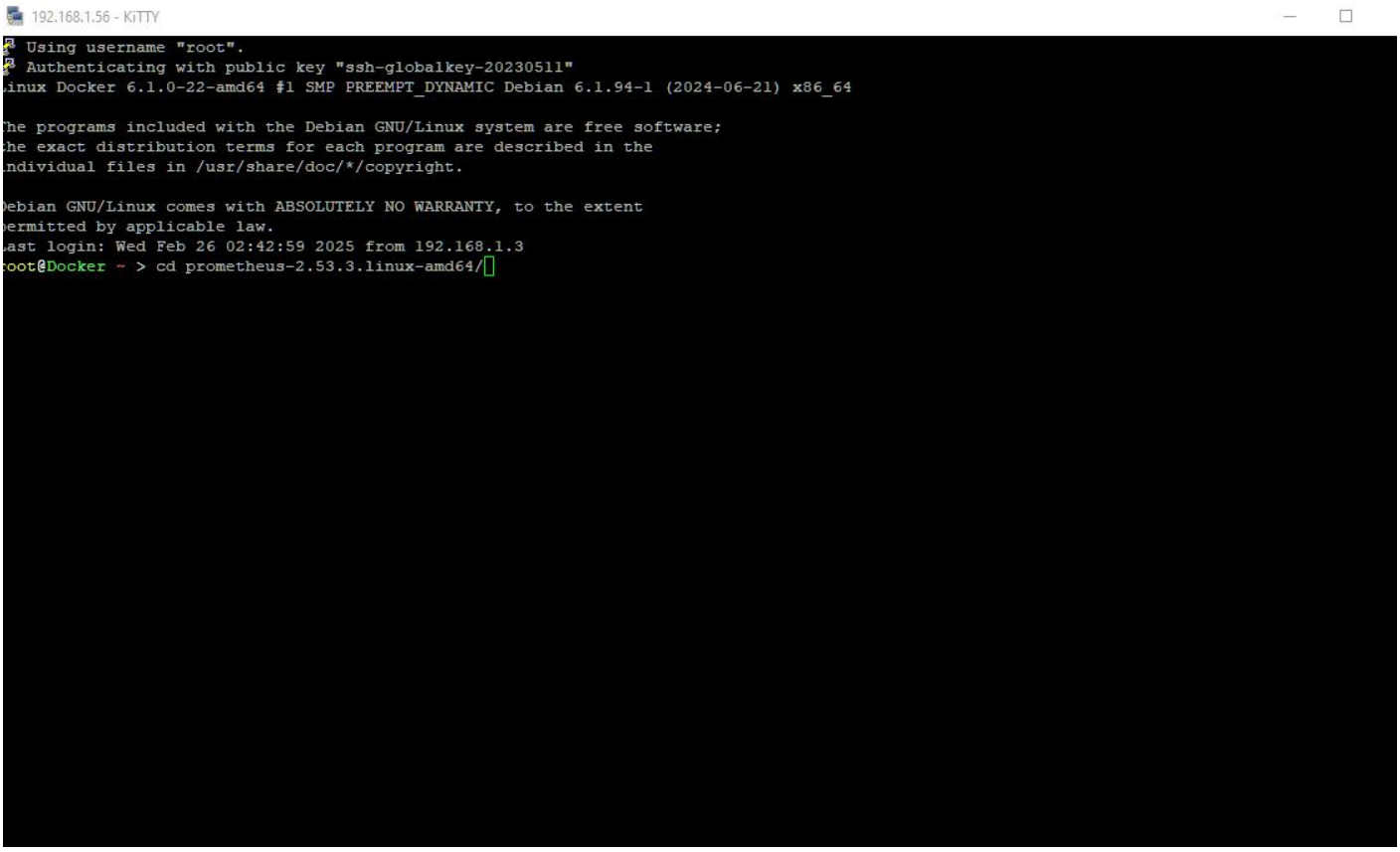
Datei entpacken:

```
tar xvfz prometheus-2.53.3.linux-amd64.tar.gz
```



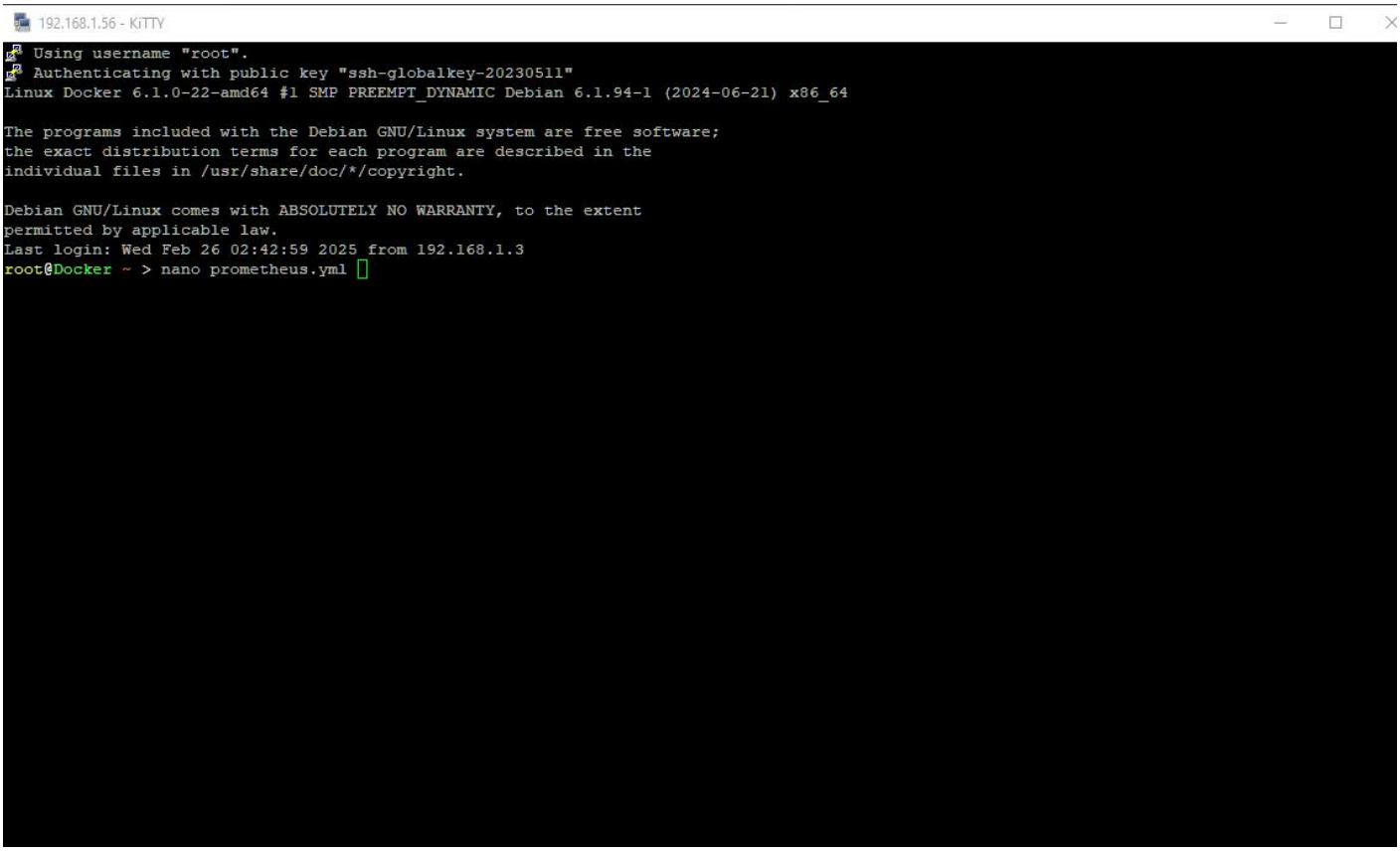
Dann ins Prometheus-Verzeichnis gewechselt:

```
cd prometheus-2.53.3.linux-amd64/
```



Dort dann die "prometheus.yml" angepasst an die URL inkl. API-Key der Mailcow (s. 2. Screenshot):

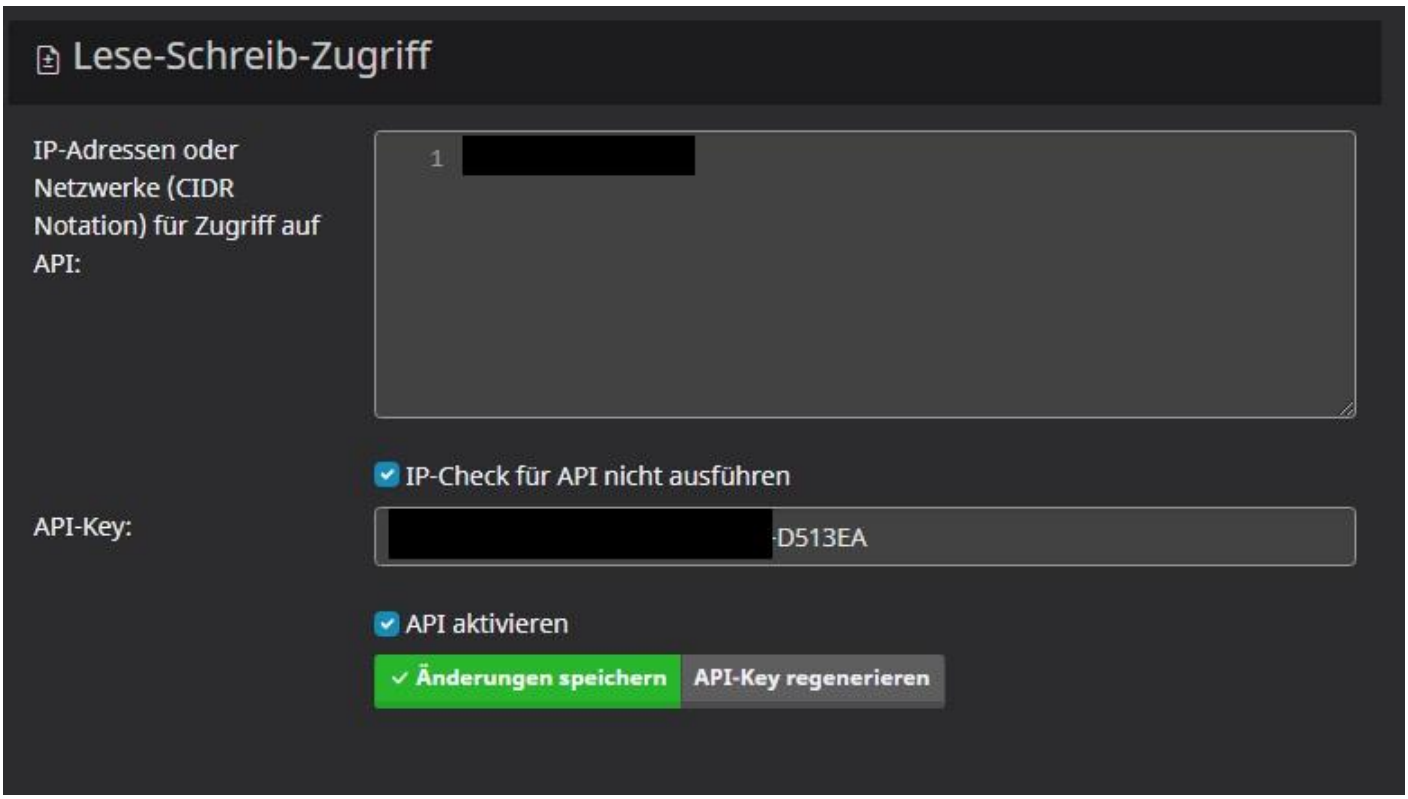
```
nano prometheus.yml
```



192.168.1.56 - KiTTY

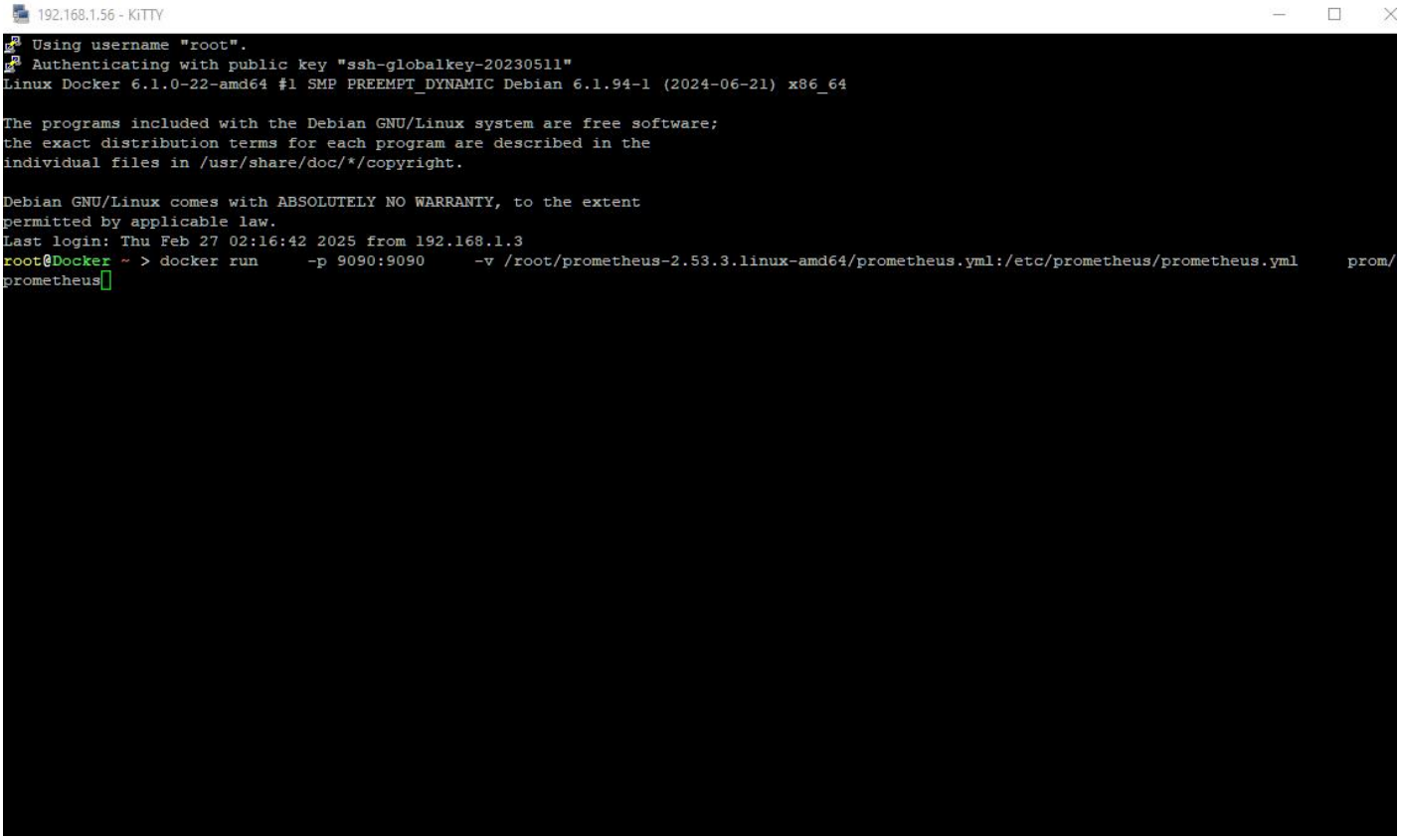
```
 GNU nano 7.2                                  prometheus.yml
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "mailcow"
    static_configs:
      - targets: [ "mail.                    .de:9099" ]
    params:
      host: [ "mail.                    .de" ]
      apiKey: [ "                       -D513EA" ]

                                [ 36 Zeilen gelesen ]
^G Hilfe      ^O Speichern  ^W Wo ist     ^K Ausschneiden^T Ausführen  ^C Position   M-U Rückgängig M-A Markierung M-] Zu Klammer
^X Beenden    ^R Datei öffnen^\ Ersetzen  ^U Einfügen   ^J Ausrichten ^/ Zu Zeile   M-E Wiederholen M-6 Kopieren  ^Q Wo war
```

Den API-Key kann man im Mailcow Config-Menü ganz einfach generieren.



Ich habe dann einmal den Container manuell mit der Config-Datei gestartet:

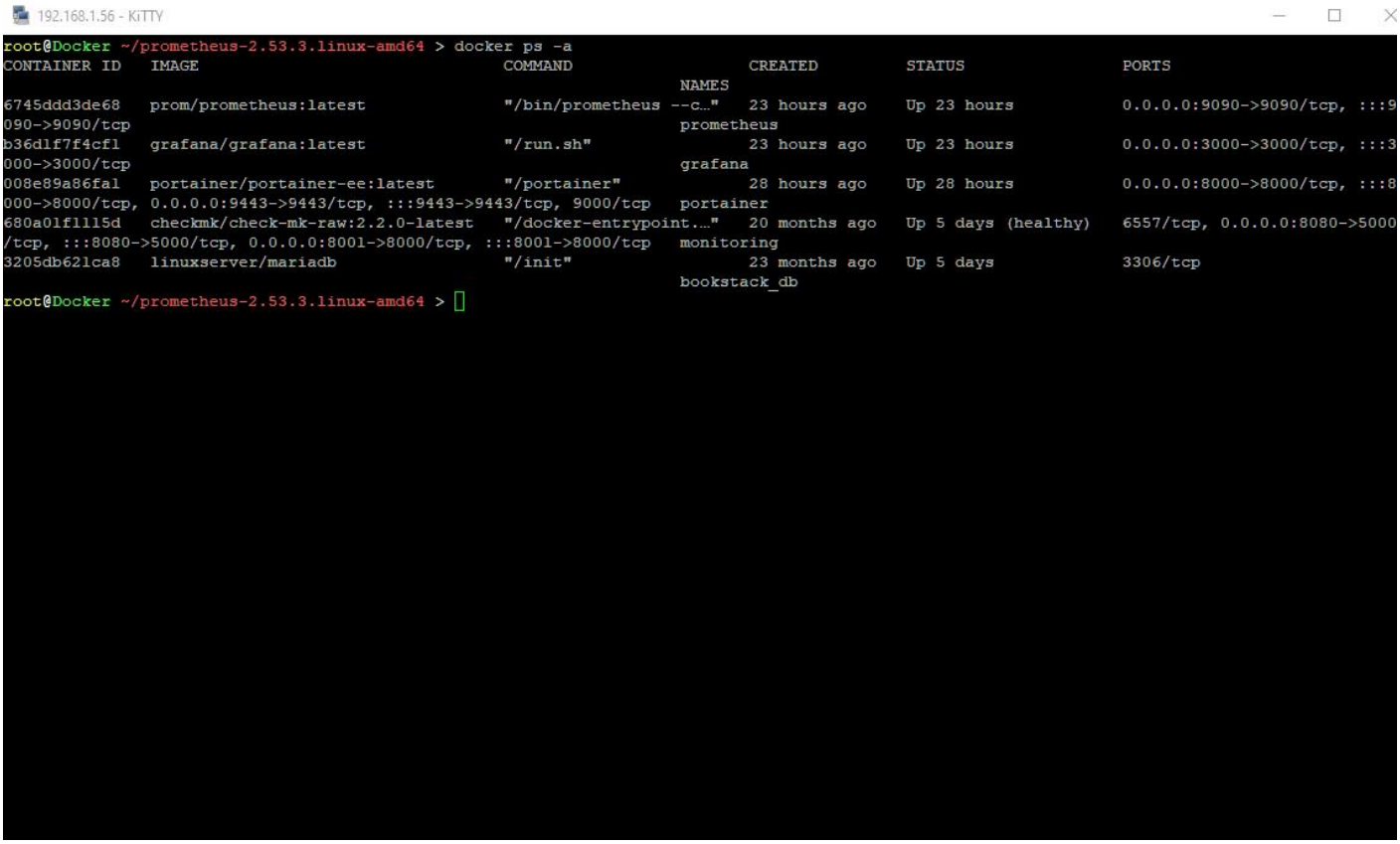
```
docker run -p 9090:9090 -v /root/prometheus-2.53.3.linux-amd64/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus
```



Um den Prometheus-Container immer automatisch zu starten, habe ich per Portainer den Prometheus-Container bei "Restart" auf "Always" gesetzt.
Alternativ geht das aber auch mit dem Befehl

```
docker run --restart always <image_name>
```

In meinem Fall heißt der Container "prom/prometheus:latest".



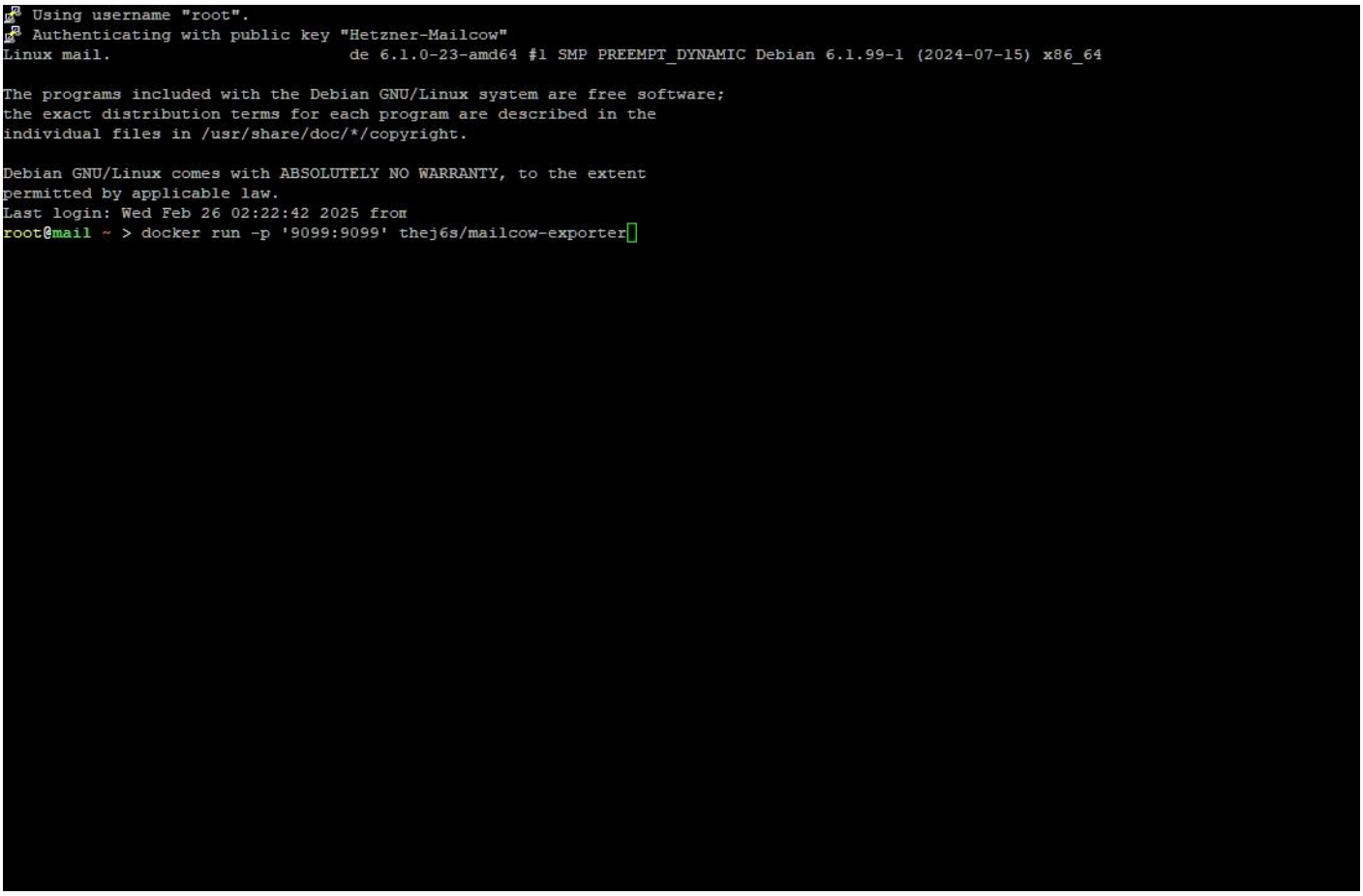
Am Besten ist es aber, über den Portainer diese Einstellung vorzunehmen, da per CLI ein neuer Container angelegt wird, warum auch immer.

Bezüglich Prometheus sind wir auf der Docker-VM fertig.

Auf der Mailcow habe ich ebenfalls Portatiner laufen. Hier müssen wir nicht so viel machen. Lediglich den Container "thej6s/mailcow-exporter" laden.

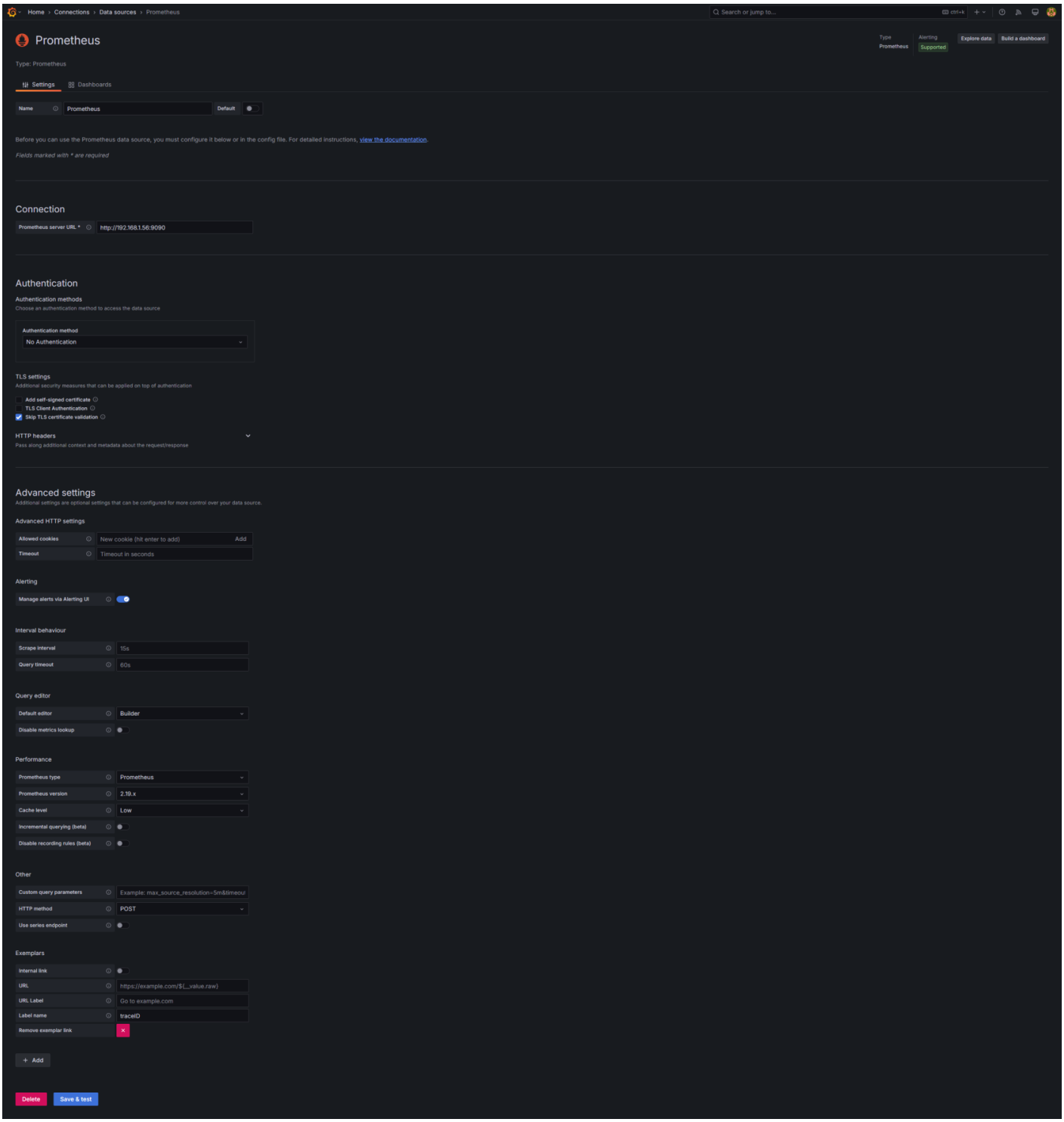
```
docker run -p '9099:9099' thej6s/mailcow-exporter
```

Auch hier habe ich per Portainer den "Restart" auf "Always" gestellt, damit dieser immer automatisch geladen wird.



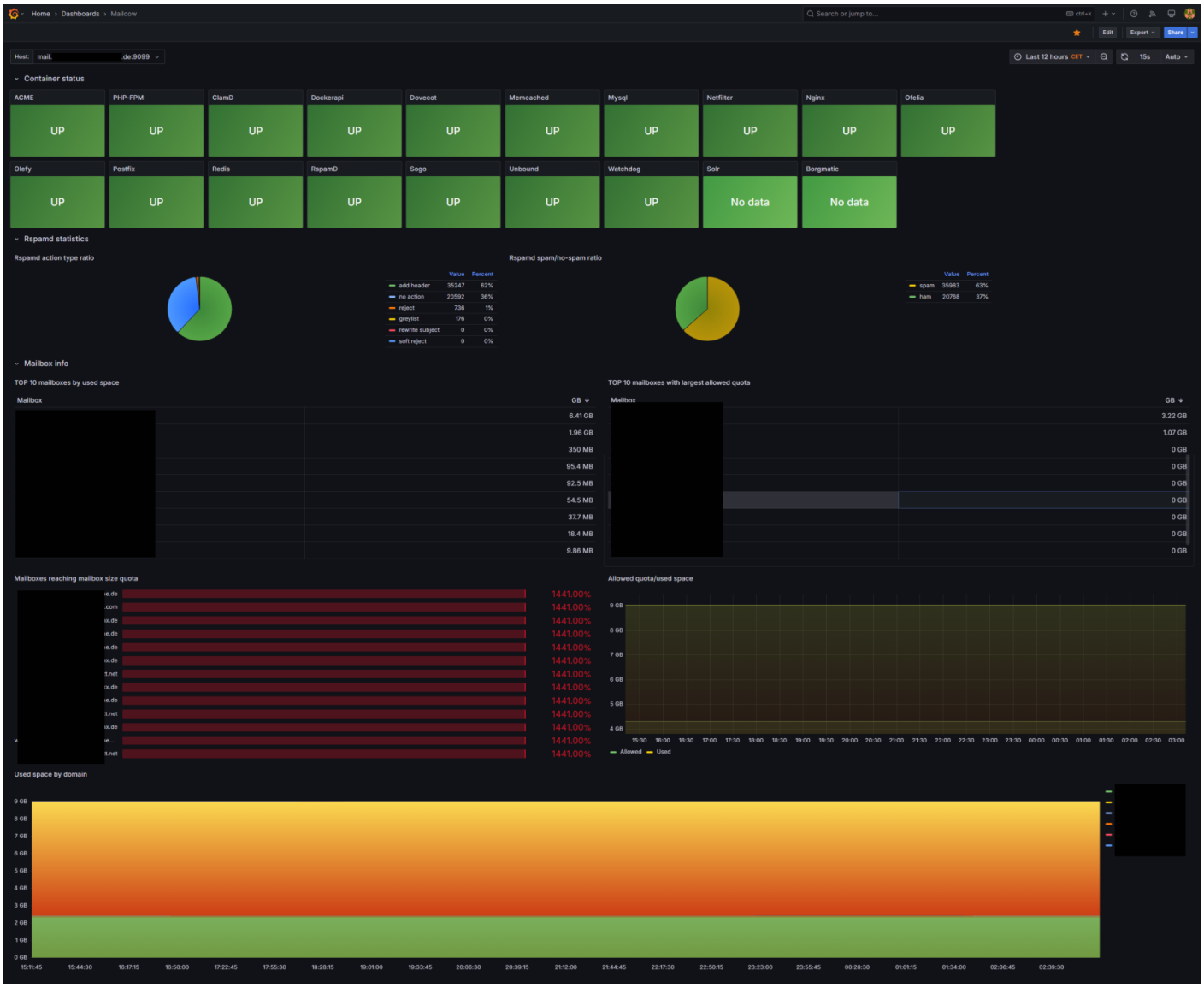
Das war's im Grunde schon.
Sobald auf der Mailcow der "thej6s/mailcow-exporter" läuft, werden die Daten schon zum Prometheus geschoben und wir können unser Dashboard im Grafana "basteln" bzw. eher gesagt "klauen". :)

Im Grafana dann noch die Datenquelle "Prometheus" hinzufügen:

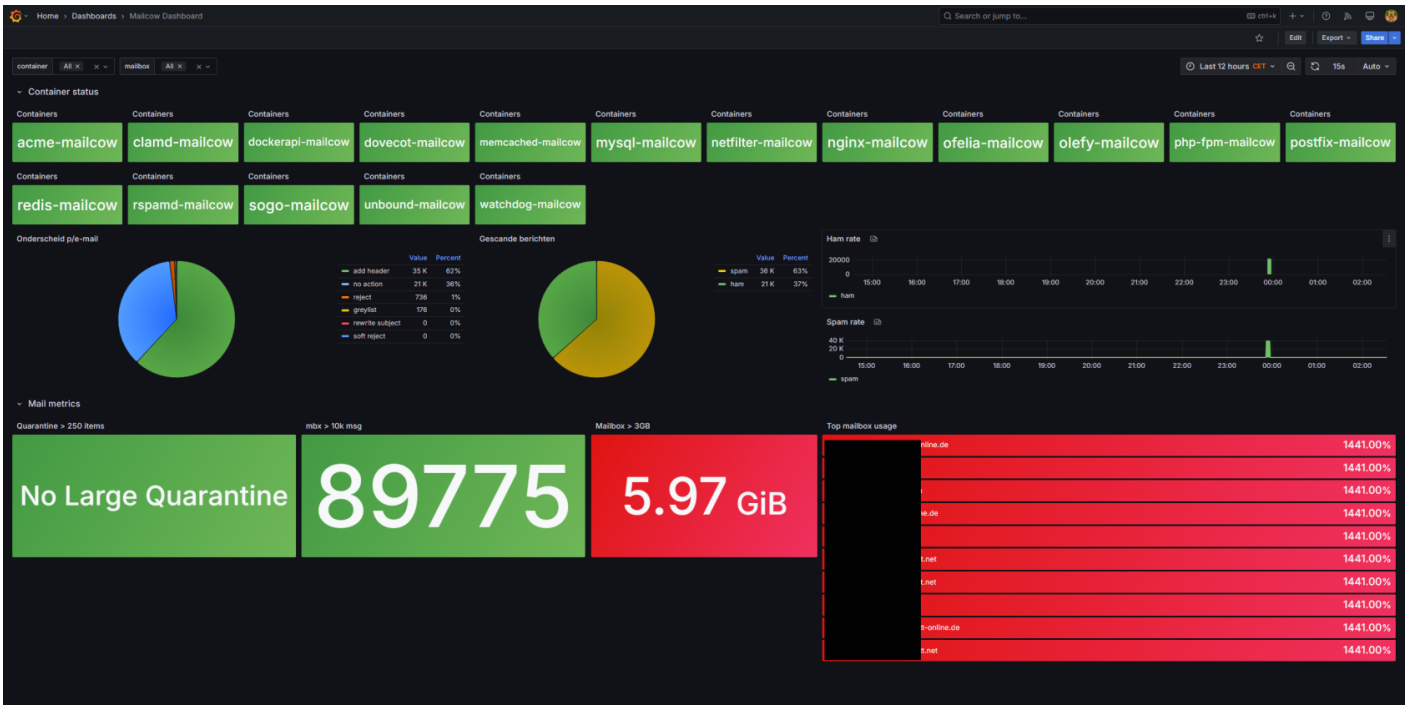


Danach können wir die Dashboards in Grafana importieren.
Ich habe diese beide Dashboards genommen (sind aber einfach nur importiert, da ist noch nichts angepasst. Daher die wilden Werte).:

https://github.com/hagaram/mailcow-grafana-dashboard



https://github.com/remkolodder/mailcow-dashboard



Fertig. :)

MailPiler-Installation

Schritt-für-Schritt-Anleitung

Mail Piler ist eine Open-Source Software zur Archivierung von E-Mails. Im Business Umfeld muss zum Beispiel jede E-Mail für 10 Jahre archiviert werden. In der folgenden Anleitung werde ich die Installation unter Rocky Linux 9 vornehmen. Da Rocky von RedHat abstammt, bietet es einen sehr langen Support und eignet sich hervorragend für Archivierungsanwendungen. Die Distro bietet noch offiziellen Support bis 2032.

Installation des OS

Als erstes wird eine Installations ISO benötigt. Ich habe meiner VM 2 Kerne, 2GB RAM und 32GB Speicher zugewiesen, diese Ressourcen können aber auch jederzeit angepasst werden. Unter Proxmox (auf einem anderen Hypervisor habe ich es noch nicht getestet) funktioniert die Installation nur, wenn der CPU Typ auf Host gestellt ist. Nachdem der Installer gebootet ist, müssen folgende Einstellungen festgelegt werden:

- Sprache Tastaturlayout (en entfernen, DE hinzufügen)
- root Passwort
- Software Selection => Minimal Install
- Zielfestplatte

Nun kann das Betriebssystem installiert werden. Nach dem ersten Starten muss das System noch auf den aktuellsten Stand gebracht werden.
Dies geschieht mit dem folgenden Befehl:

```
dnf update
```

Nun müssen die extra Packages für RHEL aktiviert werden.
Dies geschieht mit dem folgenden Befehl:

```
dnf -y install epel-release
```

Nachdem wir die extra Pakete installiert haben, geht es mit ein paar allgemeinen Tools, die auf jedem System meiner Meinung nach installiert werden sollten, weiter:

```
dnf install htop wget nano git tar
```

Nachdem dies erledigt ist, müssen wir der VM noch einen Hostnamen geben. Dazu die Datei `/etc/hostname` bearbeiten und den Namen eintragen (bspw. mit nano). Abschließend muss zum Übernehmen des Namens noch die VM neugestartet werden.

Abhängigkeiten installieren

Nachdem das Betriebssystem nun aufgesetzt ist, kann mit der eigentlichen Installation begonnen werden. Ein Teil der Abhängigkeiten kann direkt als Paket heruntergeladen werden, der Rest muss so auf dem System installiert werden.

`dnf install https://repo.manticoresearch.com/manticore-repo.noarch.rpm`
`dnf install openssl mariadb mariadb-server php php-ldap php-gd php-memcache php-pdo php-mysqli php-curl php-zip nginx manticore manticore-extra tre-devel sysstat python3 gcc poppler libytnefdevel memcached tcp_wrappers-libs openssl-devel zlib-devel mariadb-connector-c-devel`
`systemctl enable --now sysstat`
`wget http://ftp.wagner.pp.ru/pub/catdoc/catdoc-0.95.tar.gz tar`
`-xzf catdoc-0.95.tar.gz`
`cd catdoc-0.95`
`./configure`
`make`
`make install`
`cd ..`
rm catdoc-0.95 -Rf
`dnf install http://repo.iotti.biz/CentOS/9/x86_64/unrtf-0.21.9-17.el9.lux.x86_64.rpm`
http://repo.iotti.biz/CentOS/7/x86_64/xlhtml-0.5-19.el7.lux.x86_64.rpm
`https://yum.oracle.com/repo/OracleLinux/OL7/latest/x86_64/getPackage/tcp_wrappers-devel-7.6-77.el7.x86_64.rpm`
`https://yum.oracle.com/repo/OracleLinux/OL7/latest/x86_64/getPackage/tcp_wrappers-libs-7.6-77.el7.x86_64.rpm https://dl.rockylinux.org/pub/rocky/9/CRB/x86_64/os/Packages/l/libzip-devel-1.7.3-7.el9.x86_64.rpm`
`systemctl start mariadb`
`systemctl enable mariadb`
`systemctl start nginx`
`systemctl enable nginx`
`systemctl disable manticore`
`systemctl stop manticore`
`chown nginx:nginx /var/lib/php/session -R`

SELinux deaktivieren

`nano /etc/sysconfig/selinux`
`SELINUX=disabled`
`/etc/selinux/config`

`SELINUX=disabled`

Firewall Ports öffnen

`firewall-cmd --zone=public --permanent --add-port=80/tcp`
`firewall-cmd --zone=public --permanent --add-port=25/tcp`

Nun muss noch in `/etc/php-fpm.d/www.conf` der `user` und der `group` Parameter auf `nginx` geändert werden. Anschließend den PHP-FPM Prozess mit `systemctl restart php-fpm` neu starten.

Piler installieren

Nun sollten alle Abhängigkeiten auf dem System installiert sein und wir können mit der Installation des Pilers an sich weitermachen.
Dazu muss als Erstes ein neuer Nutzer angelegt werden:

`groupadd piler`
`useradd -g piler -m -s /bin/bash -d /var/piler piler`
`usermod -L piler`
`chmod 755 /var/piler`

Nachdem der Nutzer angelegt ist, können wir weiterhin als root Piler bauen und installieren:

`wget https://bitbucket.org/jsuto/piler/downloads/piler-1.4.1.tar.gz`
`tar -xzf piler-1.4.1.tar.gz`
`cd piler-1.4.1`
`./configure --localstatedir=/var --with-database=mariadb`
`make`
`make install`
`echo "/usr/local/lib" >> /etc/ld.so.conf`
`ldconfig`

Nun ist Mail Piler auf dem System installiert. Zur initialen Konfiguration müssen wir nun mit `OPENSSL_ENABLE_SHA1_SIGNATURES=1 make postinstall` den Konfigurationsassistenten durchlaufen.
Dabei müssen die folgenden Optionen eingegeben werden:

WebServer Gruppe: nginx
MySQL Hostname: localhost
MySQL Socket: /var/lib/mysql/mysql.sock
MySQL Database: bestätigen
MySQL Nutzername: bestätigen
MySQL Piler Passwort: Zufallspasswort generieren

MySQL Root Passwort: egal was, da socket Authentifizierung verwendet wird
SMTP Relay Host: öffentlicher Domain des Mail Piler
SMTP Port: 25
2x mit y bestätigen

Das Script sollte nun erfolgreich durchlaufen und zum Abschluss die Meldung `Done post installation tasks.` bringen. Nachdem der eigentliche Piler jetzt läuft, müssen wir den WebServer konfigurieren. Dazu habe ich eine fertige Konfiguration, welche wir nur einspielen müssen und anschließend den Nginx neustarten:

```
rm /etc/nginx/nginx.conf
wget -O /etc/nginx/nginx.conf https://jonasled.de/download/mailpiler-nginx.conf
systemctl restart nginx
```

Wenn alles gut gelaufen ist, sollte nun das Login über den Web Browser erreichbar sein.
Auf der Oberflöche können wir uns nun mit `admin@local` als Nutzername und `pilerrocks` als Passwort anmelden. Falls es nach dem Login zu einem Redirect auf eine falsche Domain kommt,muss in der Datei `/usr/local/etc/piler/config-site.php` der `SITE_NAME` geändert werden. Hier wird aber noch nicht viel funktionieren, da die Dienste noch nicht funktionieren. Dazu müssen zuerst die Services und einige Dateien an der richtigen Stelle verlinkt werden und danach die Services gestartet werden:

```
ln -s /usr/local/sbin/piler /usr/sbin/piler
ln -s /usr/local/sbin/piler-smtp /usr/sbin/piler-smtp
ln -s /usr/local/etc/piler /etc/piler
ln -s /usr/local/etc/piler/manticore.conf /usr/local/etc/piler/sphinx.conf
ln -s /usr/local/libexec/piler/pilersearch.service /etc/systemd/system/pilersearch.service
ln -s /usr/local/libexec/piler/piler.service /etc/systemd/system/piler.service
ln -s /usr/local/libexec/piler/piler-smtp.service /etc/systemd/system/piler-smtp.service
systemctl daemon-reload
systemctl enable --now piler
systemctl enable --now pilersearch
systemctl enable --now piler-smtp
```

Nachdem die Dienste laufen, sollte nun auch die Administrator Oberfläche funktionieren.
Hier kann auch unter administration => users ein Passwort für den auditor Nutzer festgelegt werden. Dieser darf alle E-Mails sehen. Um den Piler zu testen, können wir nun testweise eine EMail an diesen schicken. Die Empfängeradresse ist hierbei egal.

Da der Piler nicht dauerhaft die E-Mails aktualisiert, kann dies nun bis zu einer Stunde dauern, bis die Test E-Mail in der Oberfläche auftaucht.
Um dies zu beschleunigen, kann man auch die entsprechenden Befehle als Piler Nutzer ausführen:

```
# Als Piler Nutzer anmelden
su piler
```

`/usr/bin/indexer --quiet tag1 --rotate --config /usr/local/etc/piler/manticore.conf`
`/usr/bin/indexer --quiet note1 --rotate --config /usr/local/etc/piler/manticore.conf`
`/usr/local/libexec/piler/indexer.delta.sh`

Danach sollte in der Oberfläche, wenn man als Auditor Nutzer angemeldet ist, eine E-Mail erscheinen.

Login via IMAP

Damit sich auch Besitzer einer Mailbox am Piler anmelden können und auch E-Mails aus dem Piler wiederherstellen können, kann sich der Piler gegen einen IMAP Server authentifizieren. Dazu muss in der Datei `/usr/local/etc/piler/config-site.php` folgender Inhalt hinzugefügt und entsprechend angepasst werden:

```
$config['ENABLE_IMAP_AUTH'] = 1;
$config['RESTORE_OVER_IMAP'] = 1;
$config['IMAP_RESTORE_FOLDER_INBOX'] = 'INBOX';
$config['IMAP_RESTORE_FOLDER_SENT'] = 'Sent';
Login via IMAP$config['IMAP_HOST'] = '<HOST DES E-MAIL SERVERS>';
```