Monitoring: Prometheus, Loki und Grafana

Zur Überwachung der Stabilität des Systems, von Verbrauchswerten und mehr können wir verschiedene Tools verwenden. Auf dieser Seite sind Beispiele und deren Verwendung dargelegt:

- Werkzeuge zum Metriken erfassen
  - Prometheus
  - FabAccess Prometheus Exporter
  - mqtt-exporter
  - Alloy + Loki
- Grafana (visuell ansprechende Dashboards)

Klassisches Setup mit Raspberry OS (Debian 12 Bookworm)

Installation von Grafana

https://grafana.com/tutorials/install-grafana-on-raspberry-pi

```
sudo mkdir -p /etc/apt/keyrings/
wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee
/etc/apt/keyrings/grafana.gpg > /dev/null
echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a
/etc/apt/sources.list.d/grafana.list
sudo apt update
sudo apt install -y grafana
sudo /bin/systemctl enable grafana-server --now
sudo /bin/systemctl status grafana-server
```

Nach der Installation ist das Grafana Web Interface unter http://fabaccess.local:3000 erreichbar. Weitere Tipps und Tricks zur Einrichtung von Grafana sind an dieser Stelle aktuell eher Out of Scope und werden nicht behandelt.

Installation von Prometheus

Wir beziehen uns zum Teil auf die Dokumentation von https://pimylifeup.com/raspberry-pi-prometheus

Dedizierten Prometheus User hinzufügen

```
sudo useradd -m -s /bin/bash prometheus
```

Prometheus installieren. Downloads: https://github.com/prometheus/prometheus/tags

```
cd /opt
wget https://github.com/prometheus/prometheus/releases/download/v3.2.1/prometheus-3.2.1.linux-arm64.tar.gz
tar xfz prometheus-3.2.1.linux-arm64.tar.gz
mv prometheus-3.2.1.linux-arm64/ prometheus/
rm prometheus-3.2.1.linux-arm64.tar.gz

chown -R prometheus:prometheus prometheus/
```

Wir fügen etwas Web Security hinzu, da sonst jeder später den Prometheus Web Service ohne Passwort aufrufen kann. Je nach Setup kann das okay sein oder auch nicht. Wir fügen es wie folgt ein (siehe auch https://prometheus.io/docs/guides/basic-auth):

```
sudo apt install python3-bcrypt
```

```
sudo vim /opt/prometheus/gen-pass.py
```

```
import getpass
import bcrypt

password = getpass.getpass("password: ")
hashed_password = bcrypt.hashpw(password.encode("utf-8"), bcrypt.gensalt())
print(hashed_password.decode())
```

Wir führen das Script aus und geben ein Passwort ein

```
python3 /opt/prometheus/gen-pass.py
```

Den daraus gewonnenen Output nutzen wir in folgender Konfigurationsdatei, die Benutzername und Passwort enthält:

```
sudo vim /opt/prometheus/web.yml
```

```
basic_auth_users:
    admin: $2b$12$hNf2lSsxfm0.i4a.1kVpSOVyBCfIB51VRjgBUyv6kdnyTlgWj81Ay
```

Service erstellen und Prometheus starten

```
sudo vim /etc/systemd/system/prometheus.service
```

```
[Unit]
Description=Prometheus Server
Documentation=https://prometheus.io/docs/introduction/overview/
After=network-online.target

[Service]
User=prometheus
Restart=on-failure

ExecStart=/opt/prometheus/prometheus --web.config.file=/opt/prometheus/web.yml --
config.file=/opt/prometheus/prometheus.yml --storage.tsdb.path=/opt/prometheus/data

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

```
sudo systemctl daemon-reload
sudo systemctl enable prometheus.service --now
sudo systemctl status prometheus.service
```

Nach dem Start ist Prometheus erreichbar unter http://fabaccess.local:9090.

Installation von FabAccess Prometheus Exporter (Port 9000)

FabAccess hat einen eigenen Exporter für Prometheus. Dieser findet sich unter https://gitlab.com/fabinfra/fabaccess/prometheus-exporter. Der Exporter verwendet die pyfabapi (Python API Wrapper für FabAccess-API), um auf die Resourcenliste (Maschinen) und deren Zustände und Metainformationen (Namen, Kategorien) zuzugreifen. Anschließend werden diese Informationen zu passend formatierten Metriken übergeben.

Eine Beispielzeile aus unserem Demo-Setup http://fabaccess:9000/metrics:

```
bffh_machine_state{category="Central Stairs",machine_id="zam-raum1-ecke1-lamp",machine_name="1 Lampe"} 1.0
```

> Der FabAccess Exporter für Prometheus funktioniert nur mit pycapnp Version 1.3.0 oder niedriger. Ab Version 2.0.0 gibt es Fehler, die den Start des Service verhindern.
>
> Details

```
sudo apt install python3-pip python3-venv

cd /opt/prometheus/
git clone https://gitlab.com/fabinfra/fabaccess/prometheus-exporter.git fabaccess-exporter --recursive
cd /opt/prometheus/fabaccess-exporter/
python3 -m venv env
. env/bin/activate #activate venv
pip install -r requirements.txt

chown -R prometheus:prometheus /opt/prometheus/fabaccess-exporter/
```

als Service anlegen und starten

> Die Variablen `BFFH_USER` und `BFFH_PASSWORD` können mit einem beliebigen Nutzer aus BFFH befüllt werden. Sinnvollerweise hat der verwendete Nutzer mindestens globale Leserechte auf allen Ressourcen. Hierzu kann der Admin-User verwendet, oder ein dedizierter Monitoring-Benutzer angelegt werden. Wir verwenden im Beispiel einen eigenen Nutzer namens `fabaccess-prometheus-exporter`.

```
sudo vim /etc/systemd/system/prometheus-fabaccess-exporter.service
```

```
[Unit]
Description=Prometheus FabAccess Exporter Service
After=network.target

[Service]
Type=simple
User=prometheus
Group=root
Environment="EXPORTER_PORT=9000"
```

```
Environment="BFFH_HOST=YOUR.HOST.TLD"
Environment="BFFH_PORT=59661"
Environment="BFFH_USER=fabaccess-prometheus-exporter"
Environment="BFFH_PASSWORD=PASSWORD_OF_PROMETHEUS_USER_IN_BFF"
Environment="POLLING_INTERVAL_SECONDS=5"
ExecStart=/opt/prometheus/fabaccess-exporter/env/bin/python3 /opt/prometheus/fabaccess-exporter/main.py
Restart=always
RestartSec=5

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

```
sudo systemctl daemon-reload
sudo systemctl enable /etc/systemd/system/prometheus-fabaccess-exporter.service --now
sudo systemctl status prometheus-fabaccess-exporter.service
```

> **Sicherheitshinweis**
>
> Der Exporter ist im Browser auf dem Port 9000 via http erreichbar. Es ist je Setup zu überprüfen, ob das zu lauschende Interface z.B. nur `localhost` sein soll!

Upgrade von FabAccess Prometheus Exporter

Die Installation von fabaccess-exporter kann veralten. Über folgende Befehle können wir das Verzeichnis auf den aktuellsten Stand bringen:

```
cd /opt/prometheus/fabaccess-exporter/
git pull
git submodule update --recursive --remote
. env/bin/activate #activate venv
pip install --upgrade -r requirements.txt
chown -R prometheus:prometheus /opt/prometheus/fabaccess-exporter/
sudo systemctl restart prometheus-fabaccess-exporter.service
```

```
git submodule update --recursive --remote
```

Installation von mqtt-exporter (Port 9001)

Das Setup basiert auf https://github.com/kpetremann/mqtt-exporter

TLS Support: https://github.com/kpetremann/mqtt-exporter/pull/52 (aktuell nicht verwendet, weil alles auf dem gleichen Host)

```
sudo apt install python3-pip python3-venv

cd /opt/prometheus/
git clone https://github.com/kpetremann/mqtt-exporter.git
cd /opt/prometheus/mqtt-exporter/
python3 -m venv env
. env/bin/activate #activate venv
pip install -r requirements/base.txt
chown -R prometheus:prometheus /opt/prometheus/mqtt-exporter/
```

Manuell starten und testen

```
MQTT_ADDRESS=127.0.0.1 MQTT_PORT=1883 MQTT_USERNAME=fablabc MQTT_PASSWORD=THEPASSWORD
PROMETHEUS_PORT=9001 /opt/prometheus/mqtt-exporter/env/bin/python3 exporter.py
```

Als Service

```
sudo vim /etc/systemd/system/prometheus-mqtt-exporter.service
```

```
[Unit]
Description=Prometheus MQTT Exporter
After=network-online.target

[Service]
User=prometheus
Restart=on-failure

Environment="MQTT_ADDRESS=127.0.0.1"
Environment="MQTT_PORT=1883"
#TLS config - needs merged PR https://github.com/kpetremann/mqtt-exporter/pull/52
#Environment="MQTT_ENABLE_TLS=True"
#Environment="MQTT_TLS_NO_VERIFY=False"
#Environment="MQTT_ADDRESS=YOUR.HOST.TLD"
#Environment="MQTT_PORT=8883"
```

```
Environment="MQTT_USERNAME=fablabc"
Environment="MQTT_PASSWORD=THE_PASSWORD"
Environment="PROMETHEUS_PORT=9001"
ExecStart=/opt/prometheus/mqtt-exporter/env/bin/python3 /opt/prometheus/mqtt-
exporter/exporter.py

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

```
sudo systemctl daemon-reload
sudo systemctl enable /etc/systemd/system/prometheus-mqtt-exporter.service --now
sudo journalctl -f -u prometheus-mqtt-exporter.service
```

Der Log Output (Klicken zum Anzeigen):

```
PORT=9001 python3 exporter.py
INFO:mqtt-exporter:subscribing to "#"
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ty', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_if', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ofln', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_onln', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_state_0',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_state_1',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_0', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_1', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_2', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_3', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_4', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_5', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_6', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_7', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_8', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_9', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_10', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_11', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_12', labels=())
```

```
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_13', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_14', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_15', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_16', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_17', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_18', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_19', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_20', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_21', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_22', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_23', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_24', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_25', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_26', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_27', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_28', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_29', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_30', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_rl_31', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_0', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_1', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_2', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_3', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_4', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_5', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_6', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_7', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_8', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_9', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_10', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_11', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_12', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_13', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_14', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_15', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_16', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_17', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_18', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_19', labels=())
```

```
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_20', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_21', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_22', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_23', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_24', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_25', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_26', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_swc_27', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_0', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_1', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_2', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_3', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_4', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_5', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_6', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_7', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_8', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_9', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_10', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_11', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_12', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_13', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_14', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_15', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_16', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_17', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_18', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_19', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_20', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_21', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_22', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_23', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_24', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_25', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_26', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_27', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_28', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_29', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_30', labels=())
```

```
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_btn_31', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_4', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_11', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_13', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_17', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_20', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_30', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_68', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_73', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_82', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_114', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_so_117', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_lk', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_lt_st', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_bat', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_dslp', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ver', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_sn_ENERGY_Total',
labels=())
INFO:mqtt-exporter:creating prometheus metric:
PromMetricId(name='mqtt_sn_ENERGY_Yesterday', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_sn_ENERGY_Today',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_sn_ENERGY_Power',
labels=())
INFO:mqtt-exporter:creating prometheus metric:
PromMetricId(name='mqtt_sn_ENERGY_ApparentPower', labels=())
INFO:mqtt-exporter:creating prometheus metric:
PromMetricId(name='mqtt_sn_ENERGY_ReactivePower', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_sn_ENERGY_Factor',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_sn_ENERGY_Voltage',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_sn_ENERGY_Current',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_POWER', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_UptimeSec',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_Heap', labels=())
```

```
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_Sleep', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_LoadAvg',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_MqttCount',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_Wifi_AP',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_Wifi_Channel',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_Wifi_RSSI',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_Wifi_Signal',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_Wifi_LinkCount',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Total',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Yesterday',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Today',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Period',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Power',
labels=())
INFO:mqtt-exporter:creating prometheus metric:
PromMetricId(name='mqtt_ENERGY_ApparentPower', labels=())
INFO:mqtt-exporter:creating prometheus metric:
PromMetricId(name='mqtt_ENERGY_ReactivePower', labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Factor',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Voltage',
labels=())
INFO:mqtt-exporter:creating prometheus metric: PromMetricId(name='mqtt_ENERGY_Current',
labels=())
```

> **Sicherheitshinweis**
>
> Der Exporter ist im Browser auf dem Port 9001 via http erreichbar. Es ist je Setup zu überprüfen, ob das zu lauschende Interface z.B. nur `localhost` sein soll!

Upgrade von mqtt-exporter

Die Installation von mqtt-exporter kann veralten. Über folgende Befehle können wir das Verzeichnis auf den aktuellsten Stand bringen:

```
cd /opt/prometheus/mqtt-exporter/
git pull
. env/bin/activate #activate venv
pip install --upgrade -r requirements/base.txt
chown -R prometheus:prometheus /opt/prometheus/mqtt-exporter/
sudo systemctl restart prometheus-mqtt-exporter.service
```

```
git submodule update --recursive --remote
```

Prometheus Konfiguration ergänzen

Damit die beiden Exporter (mqtt-exporter und fabaccess-exporter) Daten liefern und diese dann durch Grafana grafisch ausgewertet werden können, benötigen wir eine angepasste Konfiguration.

> **Achtung:** Nicht vergessen die Basic Auth Informationen (admin:prometheus) ebenfalls einzutragen!

```
sudo vim /opt/prometheus/prometheus.yml
```

```
global:
  scrape_interval:     15s
  evaluation_interval: 15s

alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093
```

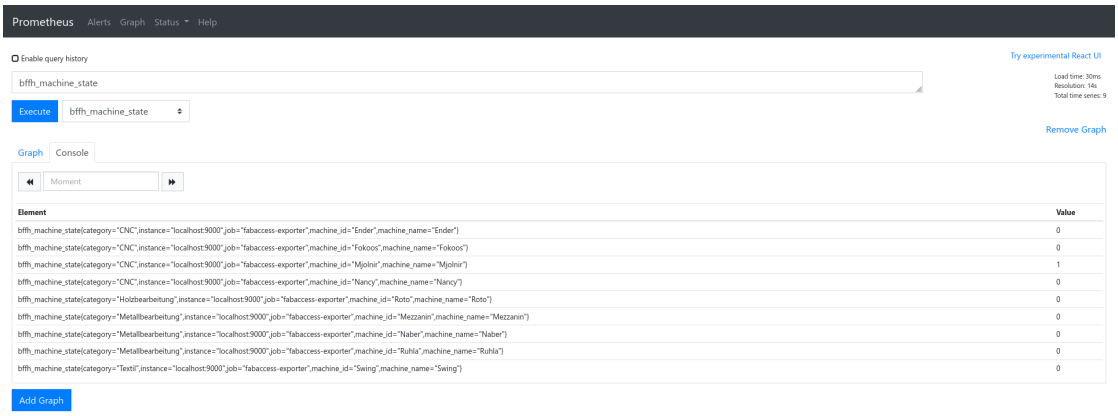
```
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
    basic_auth:
      username: 'admin'
      password: 'prometheus'
  - job_name: 'fabaccess-exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9000']
  - job_name: 'mqtt-exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9001']
```

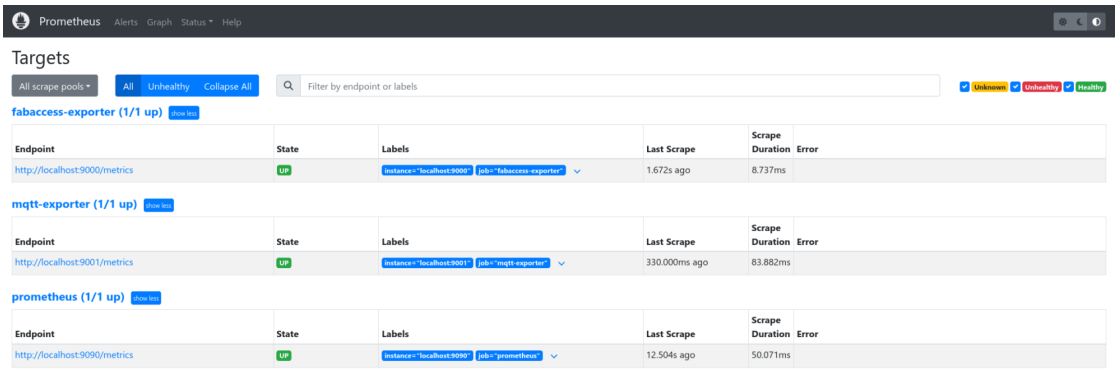
```
sudo systemctl restart prometheus.service
```

Prometheus Web Oberfläche

Beispiel Screenshot



Ob unsere Services korrekt laufen, können wir hier auch schnell überprüfen:



Installation von Alloy + Loki

Für die grafische Aufbereitung unseres Audit Logs können wir Grafana Alloy mit Grafana Loki verwenden, um daraus entsprechende Grafana Dashboards zu bauen. Wir benötigen dafür beide Komponenten.

> Achtung: Das Parsen des Audit Logs ist nur so lange zuverlässig, wie das Log nicht gelöscht oder rotiert wird. Für eine Langzeitarchivierung und eine tiefergehende Analyse von Statistiken sollte u.U. in Betracht gezogen werden den Audit in einer Datenbank zu speichern, z.B. PostgreSQL oder MariaDB.

```
sudo apt install alloy
sudo apt install loki
```

Wir haben in dieser Doku `loki` mit Version `3.4.2` und `alloy` mit `1.7.4` installiert.

Hierbei werden zwei neue Dienste installiert (Alloy auf Port 12345 und Loki auf Port 3010 per http, sowie auf Port 9096 per GRPC):

```
sudo systemctl status alloy.service
sudo systemctl status loki.service
```

Wir passen die Alloy-Konfiguration an und fügen einen passenden Job ein, um unsere Datei `audit.json` zu parsen:

```
sudo vim /etc/alloy/config.alloy
```

```
// Sample config for Alloy.
//
// For a full configuration reference, see https://grafana.com/docs/alloy
logging {
  level = "debug"
}

prometheus.exporter.unix "default" {
  include_exporter_metrics = true
  disable_collectors       = ["mdadm"]
}

prometheus.scrape "default" {
  targets = array.concat(
    prometheus.exporter.unix.default.targets,
    [{
      // Self-collect metrics
      job         = "alloy",
      __address__ = "127.0.0.1:12345",
    }],
  )

  forward_to = [
  // TODO: components to forward metrics to (like prometheus.remote_write or
  // prometheus.relabel).
  ]
}

local.file_match "bffhaudit" {
□path_targets = [{
□□__address__ = "localhost",
□□__path__    = "/var/log/bffh/audit.json",
□□job         = "bffhaudit",
□}]
}

loki.process "bffhaudit" {
□forward_to = [loki.write.default.receiver]

□stage.json {
```

```
  expressions = {
    machine   = "machine",
    state     = "state",
    timestamp = "timestamp",
  }
 }

 stage.timestamp {
  source = "timestamp"
  format = "RFC3339"
 }

 stage.output {
  source = "content"
 }
}

loki.source.file "bffhaudit" {
 targets               = local.file_match.bffhaudit.targets
 forward_to            = [loki.process.bffhaudit.receiver]
 legacy_positions_file = "/tmp/positions.yaml"
}

loki.write "default" {
 endpoint {
  url = "http://localhost:3100/loki/api/v1/push"
 }
 external_labels = {}
}
```

Wir geben Alloy den Zugriff auf unser Logfile, indem wir den Nutzer `bffh` zur Gruppe `alloy` hinzufügen:

```
groupmod -a -U alloy bffh
```

Zuletzt erstelllen bzw. passen wir die Loki-Konfiguration an. Wie lange heben wir dabei die Daten auf? Standardmäßig leben einmal durch Loki gesammelte Daten **für immer**. Das kann in Ordnung sein, jedoch unter Umständen zu kritischem Systemressourcenverbrauch (Speicherplatz) führen oder auch ein Datenschutzproblem darstellen. Je nach Space kann es hier verschiedene Bedürfnisse geben. Beim Audit Log empfehlen wir z.B. max. 2 bis 12

Monate Speicherung. Bitte auch die Konfiguration von logrotate beachten! Wir haben deshalb in die folgende Konfiguration eine Standardpolicy eingebaut, die alle 10 Minuten (`compaction_interval`) nach Logeinträgen sucht, die älter als 60 Tage sind (`retention_period`).

```
sudo vim /etc/loki/config.yml
```

```
auth_enabled: false

server:
  http_listen_address: 127.0.0.1
  http_listen_port: 3100
  grpc_listen_address: 127.0.0.1
  grpc_listen_port: 9096
  log_level: warn
  grpc_server_max_concurrent_streams: 1000

common:
  instance_addr: 127.0.0.1
  path_prefix: /tmp/loki
  storage:
    filesystem:
      chunks_directory: /tmp/loki/chunks
      rules_directory: /tmp/loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory

query_range:
  results_cache:
    cache:
      embedded_cache:
        enabled: true
        max_size_mb: 100

limits_config:
  metric_aggregation_enabled: true
```

```
schema_config:
  configs:
    - from: 2020-10-24
      store: tsdb
      object_store: filesystem
      schema: v13
      index:
        prefix: index_
        period: 24h

pattern_ingester:
  enabled: true
  metric_aggregation:
    loki_address: localhost:3100

ruler:
  alertmanager_url: http://localhost:9093

frontend:
  encoding: protobuf

analytics:
  reporting_enabled: false
```

Wenn Loki korrekt läuft, erhalten wir positiven Status per `curl` zurück:

```
curl localhost:3100/ready

# sollte zurückgeben:
ready

# oder kurz nach dem Start:
Ingester not ready: waiting for 15s after being ready
```

> **Sicherheitshinweis**
> Loki hat zwar kein Web Interface, ist jedoch über Schnittstellen auf den Ports 3100 und 9096 erreichbar. Es ist je Setup zu überprüfen, ob das zu lauschende Interface z.B. nur `localhost` sein soll!

Grafana Monitoring Dashboard

Ein FabAccess Grafana Dashboard kann unter

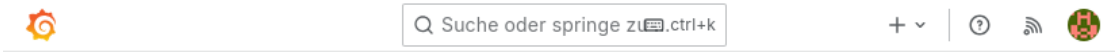
https://grafana.com/grafana/dashboards/22385 heruntergeladen werden.

Datenquellen anlegen

Bevor wir unser Dashboard importieren, legen wir zunächst jedoch unter

http://fabaccess.local:3000/connections/datasources die notwendige Prometheus und die Loki Datenquellen ("Datasources") an.

Die Prometheus Datenquelle ist in unserem Beispiel http://localhost:9090. Sofern Basic Auth in `web.yml` konfiguriert wurde, so muss dies hier ebenso eingestellt werden.



Home › Verbindungen › Datenquellen › prometheus

prometheus

Type: Prometheus

Type: Prometheus

Alerting: Supported

Explore data | Build a dashboard

Settings | Dashboards

Name: prometheus | Default

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

*Fields marked with * are required*

Connection

Prometheus server URL *: http://localhost:9090

Authentication

Authentication methods

Choose an authentication method to access the data source

No Authentication

TLS settings

Additional security measures that can be applied on top of authentication

- [ ] Add self-signed certificate
- [ ] TLS Client Authentication
- [ ] Skip TLS certificate validation

HTTP headers

Pass along additional context and metadata about the request/response

Advanced settings

Additional settings are optional settings that can be configured for more control over your data source.

Advanced HTTP settings

Allowed cookies: New cookie (hit enter to add) Add

Timeout: Timeout in seconds

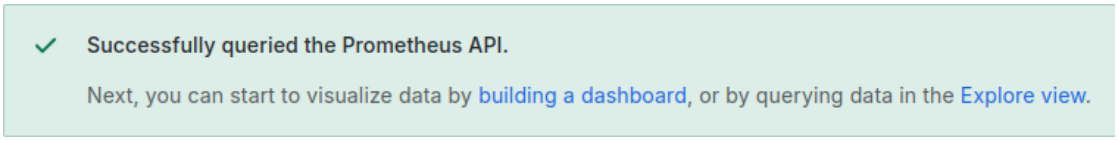
Alerting

Manage alerts via Alerting UI

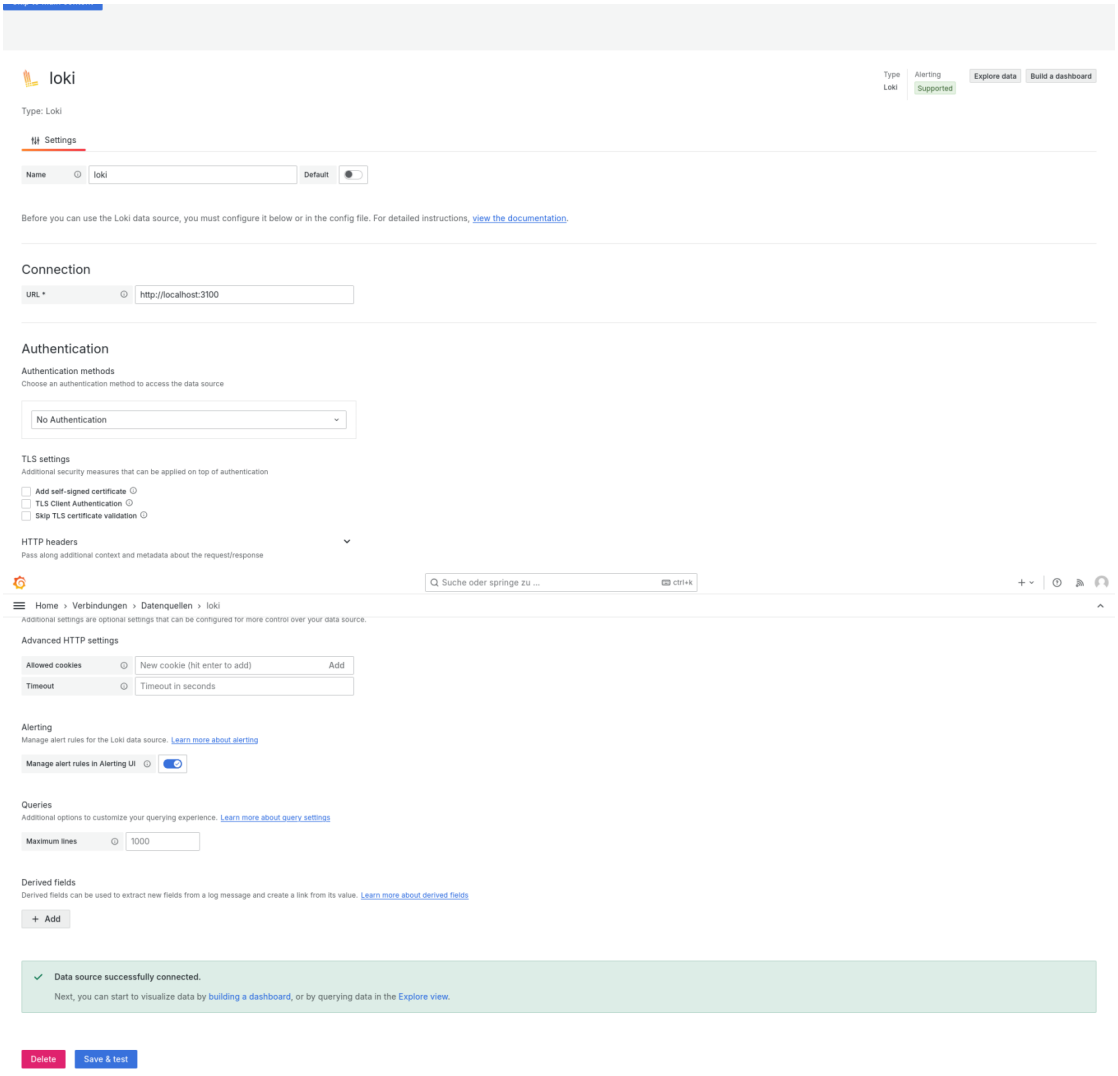
Interval behaviour

Scrape interval: 15s

Mit "Save & Test" speichern und bestätigen wir. Das Ergebnis sollte akzeptiert werden:

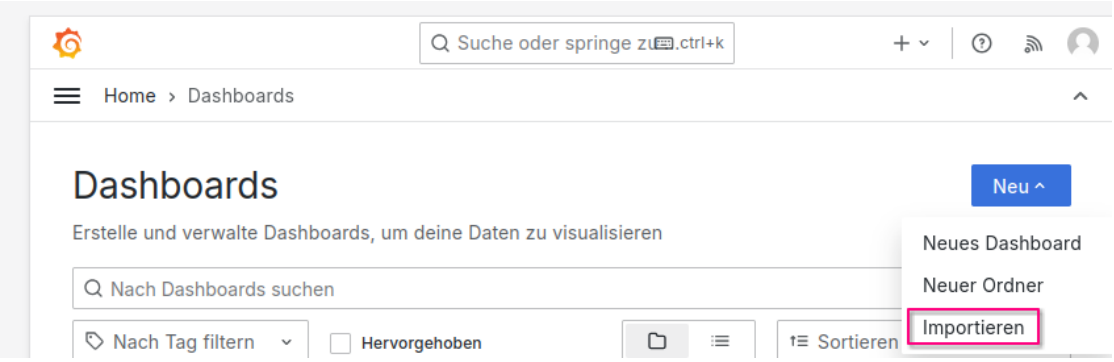


Die Loki Datenquelle ist in unserem Beispiel http://localhost:3100. Sie kann wie folgt eingebunden werden:

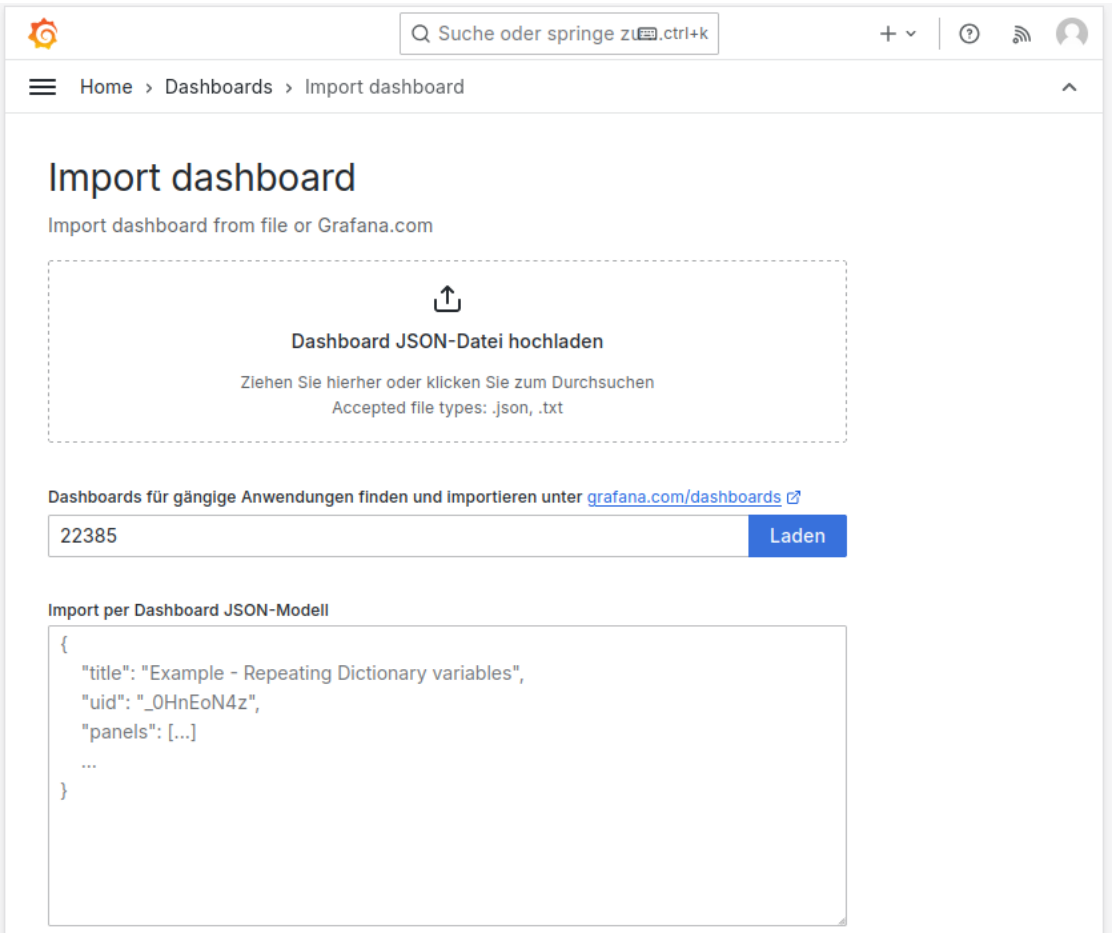


Dashboard importieren

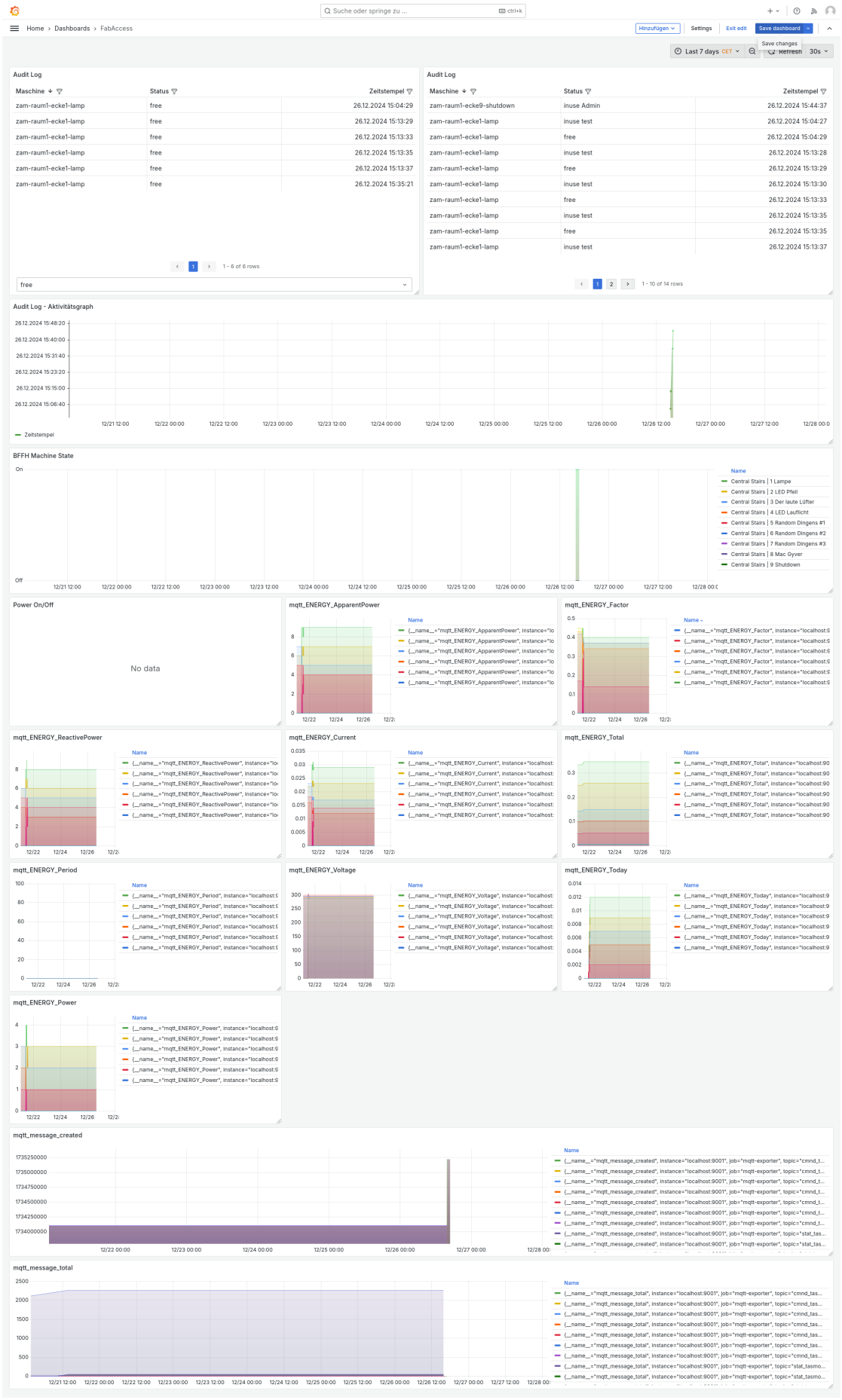
Das Importieren des Dashboards in Grafana ist sehr simpel:



Durch Eingabe der ID des Dashboards ist ein Direktimport möglich. Alternativ kann der json-Inhalt hineingeposted werden:



Beispiel Screenshot



Monitoring Setup mit Docker

Ein alternatives Setup unter Verwendung von Docker findet sich unter

https://gitlab.com/fabinfra/fabaccess/grafana

Version #61
Erstellt: 2024-10-26 12:11:23 CEST von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 2025-03-17 20:04:33 CET von Mario Voigt (Stadtfabrikanten e.V.)