Plugins und Schnittstellen

Du benötigst weitere Features, die über die Kernfunktionalitäten von FabAccess hinaus gehen? Dafür gibt es verschiedene Möglichkeiten, wie beispielsweise die Nutzung von APIs und Plugins.

- Plugin benötigt?
- Monitoring: Prometheus + Grafana
- LDAP Anbindung

Plugin benötigt?

FabAccess bietet bereits eine Vielzahl von Funktionen zur Steuerung verschiedener Geräte, jedoch sind wir stets offen für Erweiterungen und zusätzliche Funktionalitäten. Um die Integration neuer Funktionen zu erleichtern, steht ein Formular zur Verfügung, um Vorschläge für zu entwickelnde Plugins festzuhalten.

Formular

- Titel (Kurzbeschreibung des Vorschlags)
- Beschreibung (Ausführliche Beschreibung der vorgeschlagenen Erweiterung, einschließlich der Funktionalität)
- Zielgeräte/Funktionen (Welche Geräte sollen integriert oder welche Funktionen hinzugefügt werden?)
- Technische Anforderungen oder Schnittstellen
- Vorschlagende Person / Organisation (Name der vorgeschlagenen Person und gegebenenfalls die zugehörige Organisation)

Monitoring: Prometheus + Grafana

Zur Überwachung der Stabilität des Systems, von Verbrauchswerten und mehr können wir Prometheus und Grafana nutzen.

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-get update
sudo apt-get 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 nutzen aktuell Prometheus Version 2.X. Seit kurzem ist auch Prometheus in Version 3 verfügbar. Eetwaige Breaking Changes müssen noch überprüft werden.
>
> Siehe https://github.com/prometheus/prometheus/tags

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

```
cd /opt
wget https://github.com/prometheus/prometheus/releases/download/v2.55.1/prometheus-2.55.1.linux-armv7.tar.gz
tar xfz prometheus-2.55.1.linux-armv7.tar.gz
mv prometheus-2.55.1.linux-armv7/ prometheus/
rm prometheus-2.55.1.linux-armv7.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 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

#pycpnp Overwrite
pip install pycapnp==1.3.0
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!

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!

Prometheus Konfiguration ergänzen

Damit die beiden Exporter Daten liefern und diese dann durch Grafana grafisch ausgewertet werden können, benötigen wir eine angepasste Konfiguration:

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

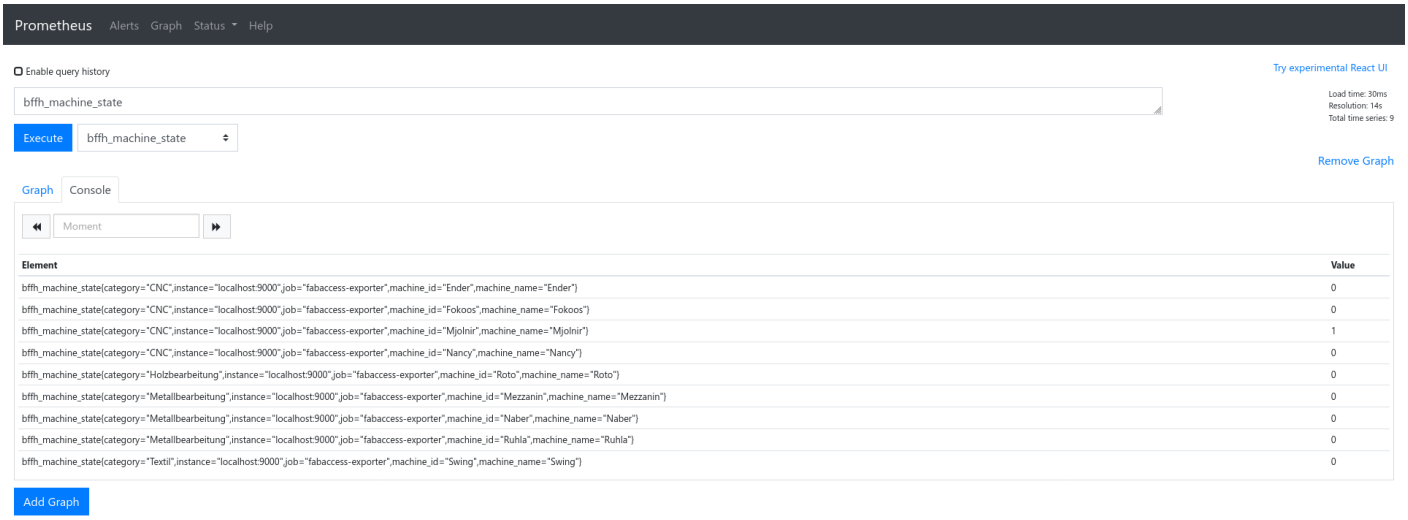
```
global:
  scrape_interval:     15s
  evaluation_interval: 15s

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']
  - job_name: 'fabaccess-exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9000']
  - job_name: 'mqtt-exporter'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9001']
```

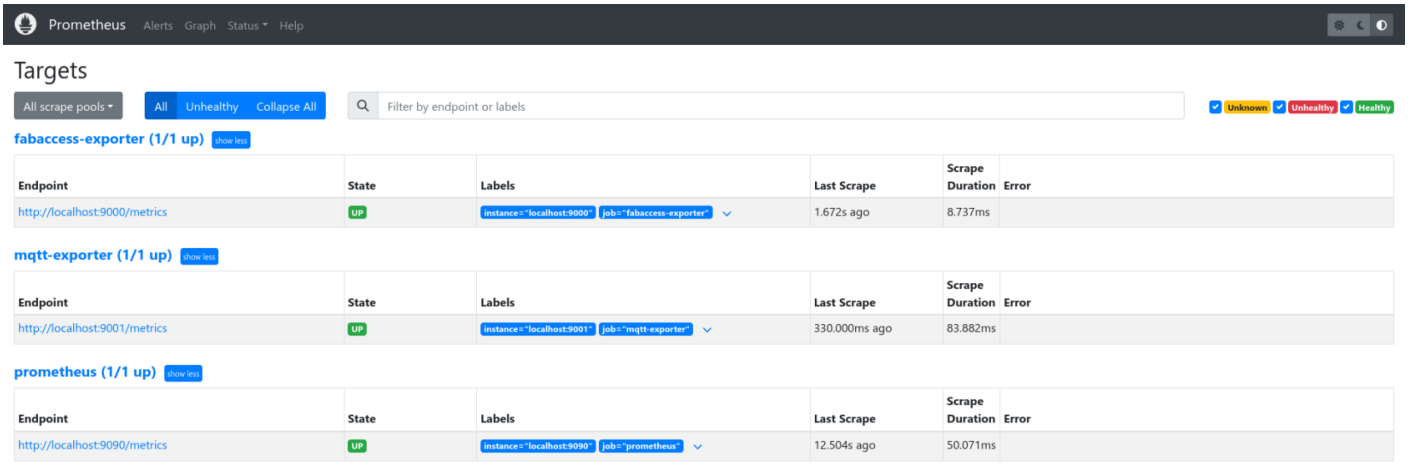
```
sudo systemctl restart prometheus.service
```

Prometheus Web Oberfläche

Beispiel Screenshot



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



Grafana Monitoring Dashboard

Ein FabAccess Grafana Dashboard kann unter

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

Datenquelle anlegen

Bevor wir es importieren, legen wir zunächst jedoch unter

http://fabaccess.local:3000/connections/datasources die notwendige Prometheus

Datenquelle ("Datasource") an. Diese ist in unserem Beispiel http://localhost:9090. Sofern

Basic Auth in `web.yml` konfiguriert wurde, so muss dies hier ebenso eingestellt werden.

Suche oder springe zu... ctrl+k

Home › Verbindungen › Datenquellen › prometheus

prometheus

Type
Prometheus

Alerting
Supported

Explore data | Build a dashboard

Type: Prometheus

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

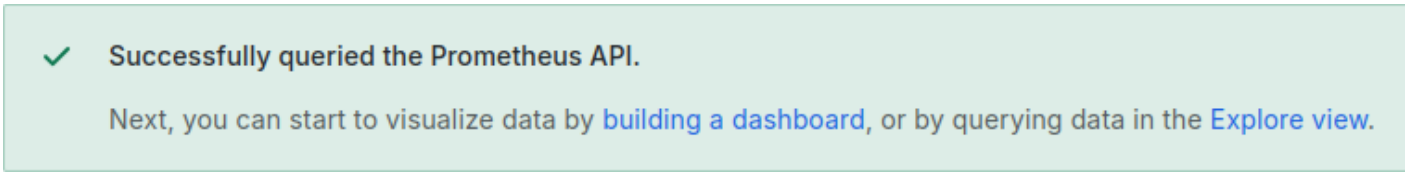
HTTP headers

Pass along additional context and metadata about the request/response

Advanced settings

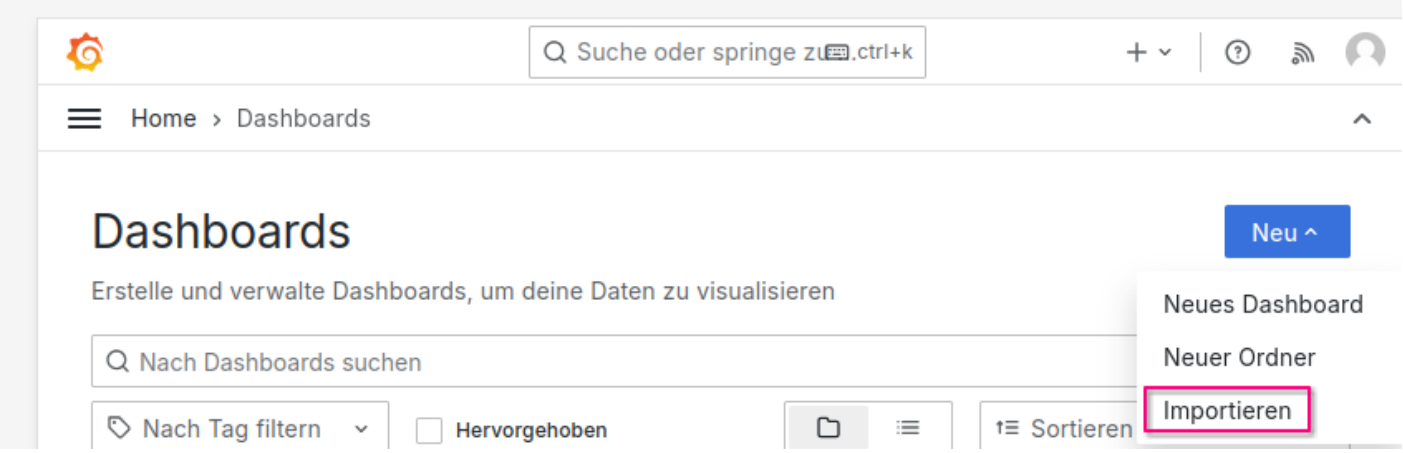
Additional settings are optional settings that can be configured for more control over your data source.

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

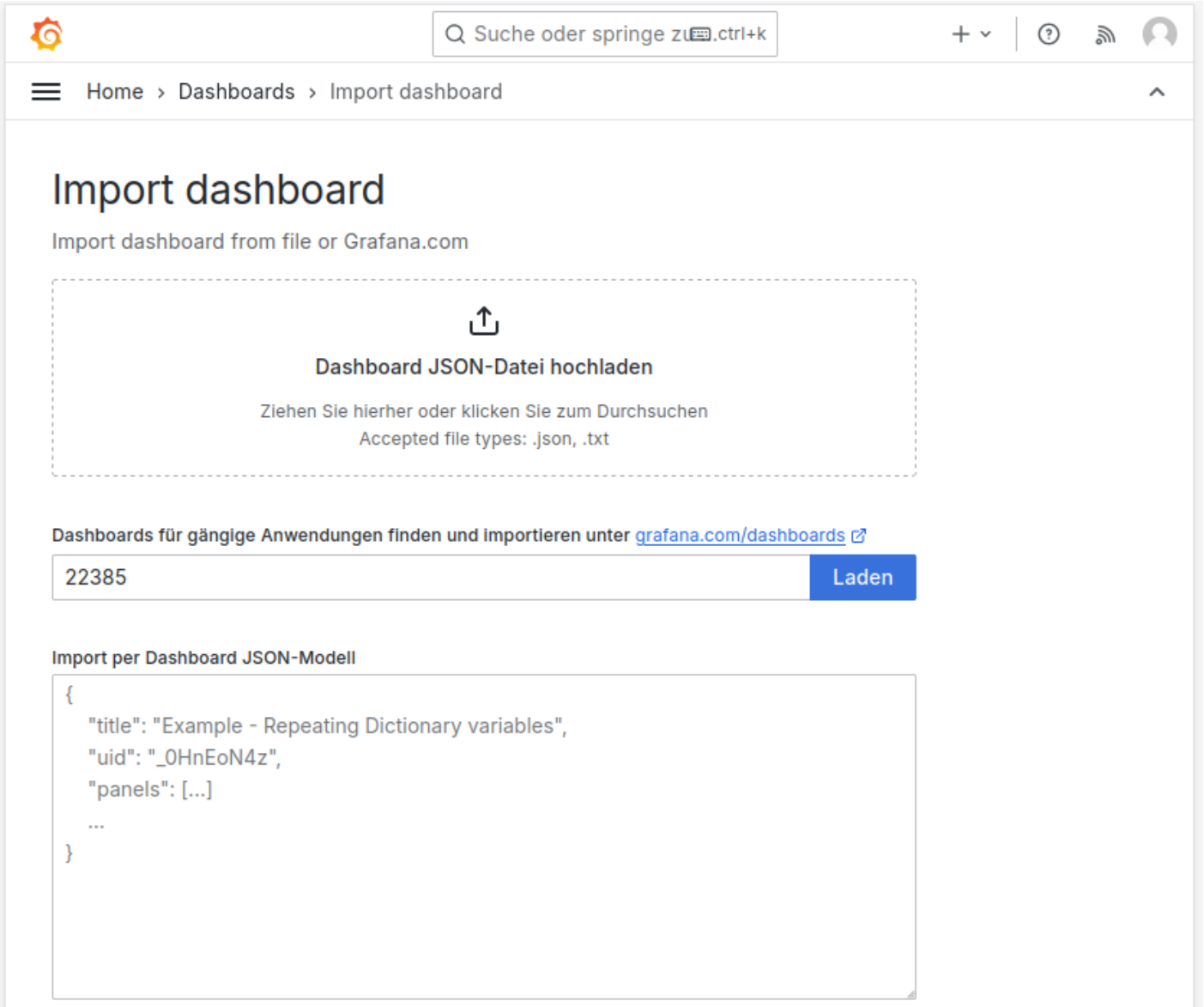


Dashboard importieren

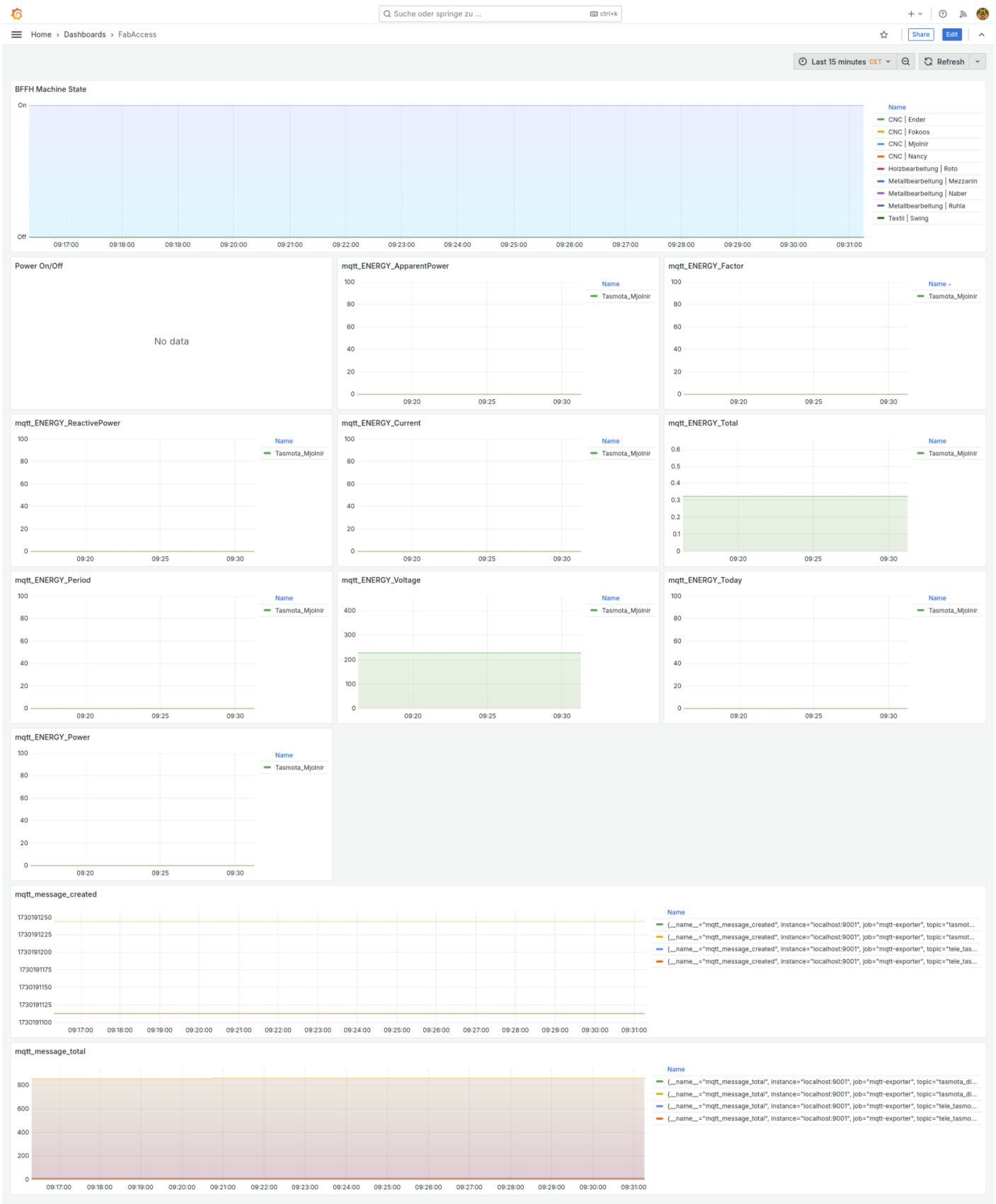
Das Importieren des Dashboards in Grafana ist sehr simpel:



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

LDAP Anbindung

LDAP ist eines der möglichen und häufig verwendeten, externen Authentifikationsverfahren.

Allgemeine Infos

FabAccess unterstützt von Haus aus derzeitig noch keine LDAP-Integration. Das liegt unter anderem daran, dass die interne Benutzerdatenbank das besondere Passwort-Hashing-Verfahren Argon2 verwendet. Argon2 wird nur von OpenLDAP Servern unterstützt. Alle anderen LDAP Server wie z.B. Synology, FreeIPA, Active Directory oder ähnlich geben hierfür keine Unterstützung.

Es besteht jedoch die Möglichkeit über ein Python-Script Nutzer in eine `users.toml` Benutzerdatendatei (siehe hier) zu exportieren und diese dann wiederrum in FabAccess zu importieren. Dieser Umweg bedeutet, dass der FabAccess Server bei etwaigen Benutzeränderungen regelmäßig neugestartet werden muss - zum Beispiel 1x täglich nachts per Cronjob oder ähnlichem. Siehe Bekannte Probleme.

FabAccess users.toml LDAP Import

Quelle: https://github.com/vmario89/fabaccess-users-toml-ldap

Zweck

Dieses Script verbindet sich bei Ausführung mit den angegebenen Credentials zu einem LDAP(S)-Server und sucht nach passenden Nutzern, um eine FabAccess-kompatible `users.toml` Datei zu erzeugen.

Das Script dient außerdem auch als Beispielvorlage für andere Entwickler, die ggf. andere Anwendungen bzw. Benutzerquellen an FabAccess anbinden wollen und nach geeigneten Code-Quellen suchen.

> **Wichtig**: Dieses Script ersetzt **keine** native LDAP-Integration in FabAccess!

Installation

```
sudo apt install build-essential python3-dev libldap2-dev libsasl2-dev ldap-utils
```

```
cd /opt/fabinfra/scripts/
git clone https://github.com/vmario89/fabaccess-users-toml-ldap.git

cd /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
chmod +x /opt/fabinfra/scripts/fabaccess-users-toml-ldap/main.py

python3 -m venv env
. env/bin/activate #activate venv
pip install -r requirements.txt

chown -R bffh:bffh /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
```

Benutzung

Hilfe / Parameter anzeigen

```
cd /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
python3 main.py --help
```

```
usage: main.py [-h] [-s SERVER] [-u USER] [-p PASSWORD] [-b BASEDN] [--filter_user FILTER_USER] [--
regex_groups REGEX_GROUPS] [--attrib_user ATTRIB_USER] [--attrib_groups ATTRIB_GROUPS]
               [--attrib_password ATTRIB_PASSWORD] [--output OUTPUT]

options:
  -h, --help            show this help message and exit
  -s SERVER, --server SERVER
                        LDAP Server (Syntax: <protocol>://host:port, e.g. ldap://192.168.1.1:389 or
ldaps://192.168.1.1.:636)
  -u USER, --user USER  User, e.g. 'uid=root,cn=users,dc=yourserver,dc=com'
  -p PASSWORD, --password PASSWORD
                        Password
  -b BASEDN, --basedn BASEDN
                        BaseDN, for example 'cn=users,dc=yourserver,dc=com'
  --filter_user FILTER_USER
                        LDAP user filter, e.g. '(&(uid=*)(objectClass=posixAccount))'
```

```
  --regex_groups REGEX_GROUPS
                        LDAP group regex, e.g. 'cn=(.*),cn=groups,dc=yourserver,dc=com'. If your group result is
'cn=administrator,cn=groups,dc=yourserver,dc=com', then the word
                        'administrator' gets properly exctracted. You can use https://regex101.com for testing.
  --attrib_user ATTRIB_USER
                        Attribute name for FabAccess user name, e.g. 'uid'
  --attrib_groups ATTRIB_GROUPS
                        Attribute name for FabAccess user roles, e.g. 'memberOf'
  --attrib_password ATTRIB_PASSWORD
                        Attribute name for FabAccess user password hash, e.g. 'sambaNTPassword'. For OpenLDAP there
is Argon2 hash support!
  --output OUTPUT       Target directory + file where to write the toml file. Please provide the full name. If not
given, users.toml will be written
```

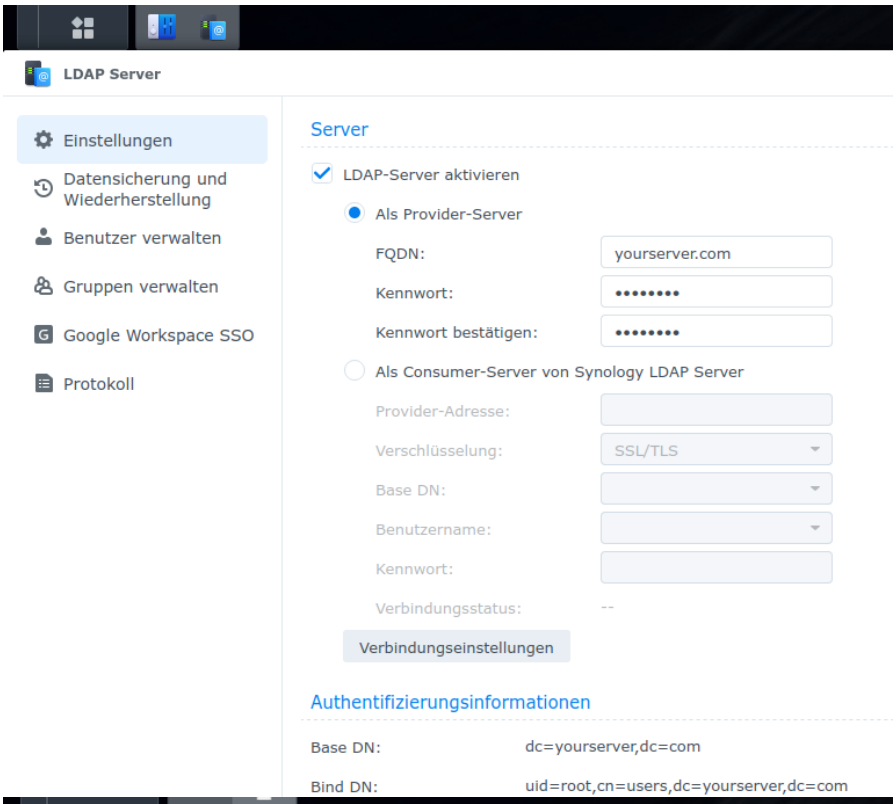
`users.toml` Datei schreiben

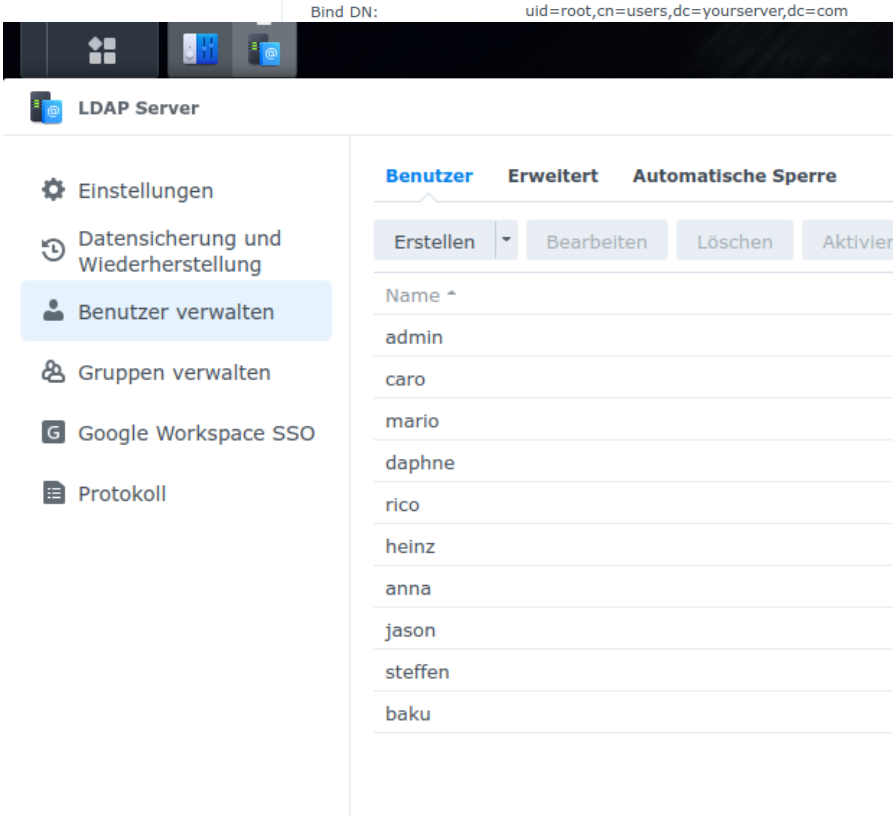
```
cd /opt/fabinfra/scripts/fabaccess-users-toml-ldap/
env/bin/python3 main.py \
  --server ldap://192.168.188.1:389 \
  --user="uid=admin,cn=users,dc=yourserver,dc=com" \
  --password pw \
  --basedn "cn=users,dc=yourserver,dc=com" \
  --filter_user "uid=*" \
  --regex_groups "cn=(.*),cn=groups,dc=yourserver,dc=com" \
  --attrib_user uid \
  --attrib_groups memberOf \
  --attrib_password sambaNTPassword \
  --output /opt/fabinfra/bffh-data/config/users.toml
```

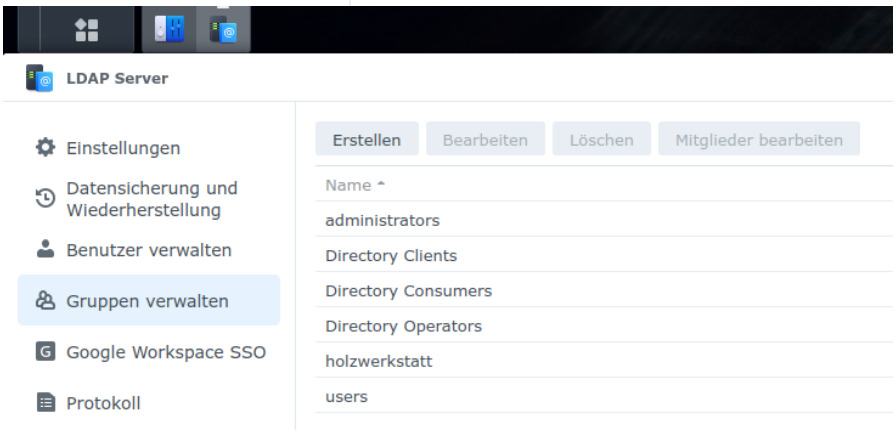
Nach dem Erstellen der Datei sollte diese überprüft und im Anschluss per `bffhd --load users.toml` geladen werden, um die Änderungen entsprechend zu reflektieren. Dafür gibt es auch geeignete Scripts.

Beispiel-Anbindung eines Synology LDAP Servers

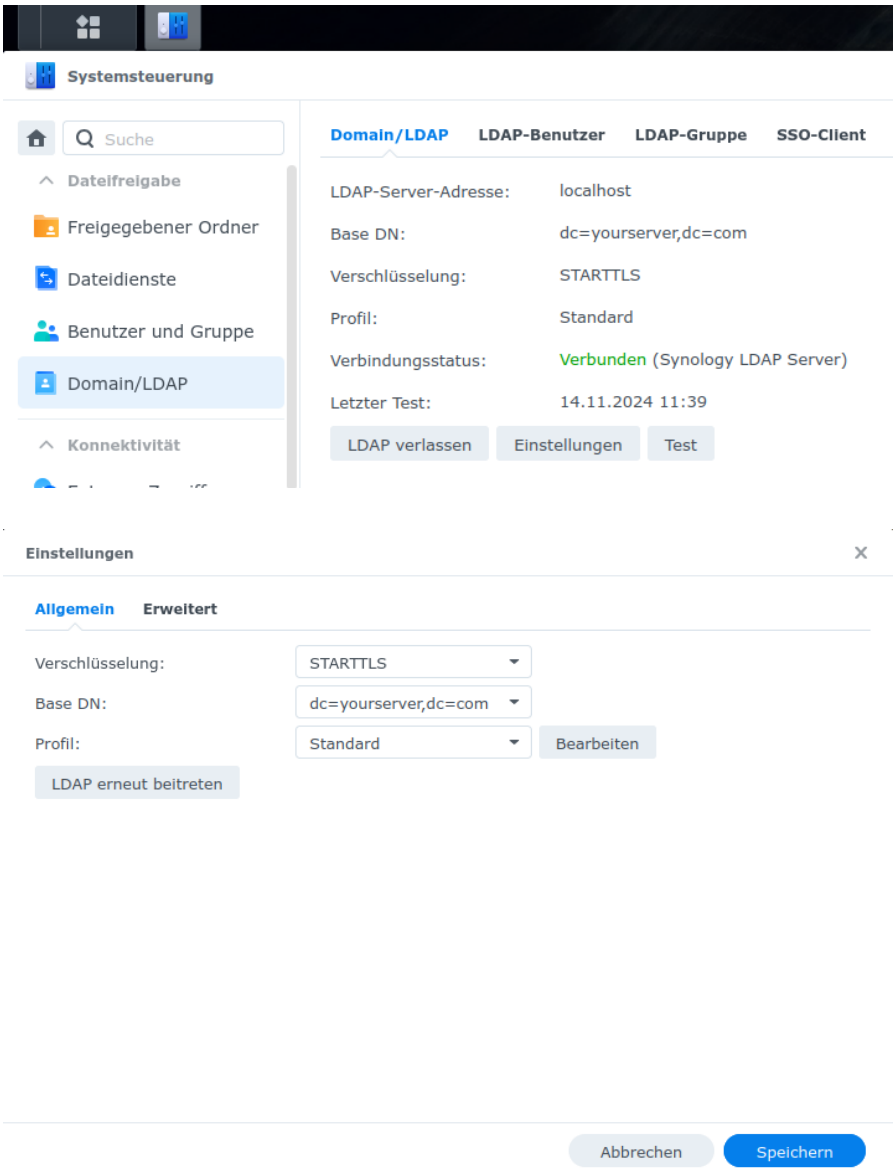
Vorraussetzung ist ein installierter und laufenden Synology LDAP Server. Hierzu auch siehe https://kb.synology.com/de-de/DSM/help/DirectoryServer/ldap_server?version=7. Dieser enthält das festgelegte Gruppen- und Benutzerschema und das entsprechende Berechtigungskonzept:


LDAP Server
Einstellungen
Datensicherung und Wiederherstellung
Benutzer verwalten
Gruppen verwalten
Google Workspace SSO
Protokoll
Server
LDAP-Server aktivieren
Als Provider-Server
FQDN:
yourserver.com
Kennwort:
Kennwort bestätigen:
Als Consumer-Server von Synology LDAP Server
Provider-Adresse:
Verschlüsselung:
SSL/TLS
Base DN:
Benutzername:
Kennwort:
Verbindungsstatus:
--
Verbindungseinstellungen
Authentifizierungsinformationen
Base DN:
dc=yourserver,dc=com
Bind DN:
uid=root,cn=users,dc=yourserver,dc=com


LDAP Server
Einstellungen
Datensicherung und Wiederherstellung
Benutzer verwalten
Gruppen verwalten
Google Workspace SSO
Protokoll
Benutzer
Erweitert
Automatische Sperre
Erstellen
Bearbeiten
Löschen
Aktivier
Name
admin
caro
mario
daphne
rico
heinz
anna
jason
steffen
baku


LDAP Server
Einstellungen
Datensicherung und Wiederherstellung
Benutzer verwalten
Gruppen verwalten
Google Workspace SSO
Protokoll
Erstellen
Bearbeiten
Löschen
Mitglieder bearbeiten
Name
administrators
Directory Clients
Directory Consumers
Directory Operators
holzwerkstatt
users

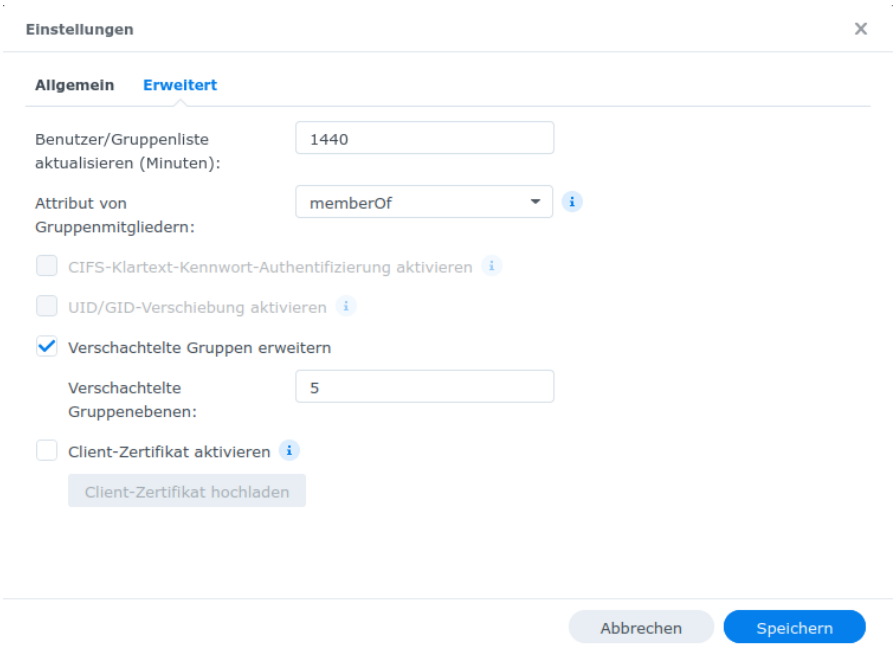
Ist dieser einmal eingerichtet, sollte die Synology NAS mit ihrem eigenen LDAP Server verbunden sein, also der Domäne beitreten:



Der Server sollte dann von allen Clients entsprechend erreichbar sein. Hierzu sind ggf. Interfaces, Port-Weiterleitungen oder die Firewall anzupassen. Wer eine sicherere Übertragung nutzen will, der benutzt das `ldaps://` Protokoll. Im Screenshot fnden wir auch auch das Attribut `--basedn` mit dem Beispielwert `dc=yourserver,dc=com`.

Das Attribut für die Benutzer (`--attrib_user`) lautet bei den meisten LDAP-Servern standardmäßig `uid` - so auch bei Synology. Unser `--user-filter` ist ein klassischer LDAP-Filter, der nach "uid" per Wildcard sucht und außerdem die Objektklasse "posixAccount" verlangt: `(&(uid=*)(objectClass=posixAccount))`. Hier muss je nach individuellem Setup herumprobiert werden, was das beste und stabilste Ergebnis wiederspiegelt.

Das gesuchte Attribut `--attrib_groups` finden wir in der Auswahlbox "Attribut von Gruppenmitgliedern:". In unserem Beispiel ist das `memberOf`.



Damit die ausgelesenen Gruppennamen nicht ellenlang werden, nutzen wir einen Regex-Filter --`regex_groups`, um nur gekürzte Rollennamen verwenden zu können. So wird aus der Zeichenkette `cn=administrator,cn=groups,dc=yourserver,dc=com` im Zusammenspiel mit dem Regex-String `cn=(.*),cn=groups,dc=yourserver,dc=com` letztlich nur das noch Gruppenwort `administrator` herausgefiltert. Für das Erstellen von `--regex_groups` kann https://regex101.com genutzt werden.

Für das Attribut --`attrib_password` können wir zum Beispiel das vom LDAP-Server bereitgestellte Passwortattribut `sambaNTPassword` verwenden. Hierzu sei gesagt, dass dies nicht hilfreich ist, da sich die Benutzer mit diesem Passworthash in FabAccess nicht anmelden können. Der Administrator muss das Passwort des Nutzers zurücksetzen. Das liegt daran, dass hier eine grundsätzliche Implementierung in bffh fehlt, die den LDAP-Server anspricht und nachfragt, ob der Nutzer in LDAP gerade existiert, die passende Berechtigung hat und die entspreche Aktion (z.B. den Login) ausführen darf bzw. das korrekte Passwort eingegeben hat. Von außen können wir den Passworthash zwar validieren, aber nicht in ein plaintext-Format umwandeln und damit also auch nicht in Argon2 umwandeln.

Bekannte Probleme

Dieses Script stellt **keine** saubere Lösung für die Nutzung von LDAP mit FabAccess bffh dar (zumindest nicht mit Version 0.4.2):

- etwaige Passwortänderungen am zentralen LDAP-Server müssen erneut in die `users.toml` Datei übertragen werden
- ändert der Nutzer oder der Administrator für den Nutzer das Passwort über die Client App, dann würde der Nutzer beim nächstens `users.toml` Import wieder

überschrieben. Diese Änderungen werden also auch nicht an den LDAP-Server gesendet
- die Password Hashes aus dem LDAP Server können in der Regel nicht mit FabAccess verwendet werden, da sie kein Argon2-Format aufweisen. Einzig OpenLDAP unterstützt Argon2 überhaupt. Aus adminstrativen Gründen macht eine Umstellung aller Nutzerpassworthashes auf Argon2 jedoch keinen Sinn. Eine native Integration von LDAP direkt in FabAccess ist also unumgänglich.

Empfehlung

Die LDAP-Gruppen sollten auf Rollen aufgebaut sein, die geeignet für die Werkstattabläufe sind. Deshalb empfehlen wir die Verwendung des FabAccess Config Generator. In diesem lassen sich Rollen geeignet definieren und die entsprechenden Maschinen in die Hauptkonfiguration schreiben. Davon abgeleitet bedarf es dann einer kompatibel gestateltenen `users.toml`.

Automatisierung

Dieses Python-Script kann zum Beispiel als Cronjob automatisiert werden, um im Zeitintervall die LDAP-Benutzer zu aktualisieren und dann den bffhd Daemon neu zu starten. Falls zum Beispiel systemd verwendet wird, könnte das wie folgt aussehen. Wir packen dabei den obigen Script-Aufruf in ein eigenständiges Bash-Script, um den Cron Job übersichtlicher zu halten:

```
vim /opt/fabinfra/scripts/fabaccess-users-toml-ldap/cron.sh
```

```
#!/bin/bash


# create a recent users.toml by connecting to LDAP Server, grabbing data
/opt/fabinfra/scripts/fabaccess-users-toml-ldap/env/bin/python3 /opt/fabinfra/scripts/fabaccess-users-toml-ldap/main.py \
  --server ldap://192.168.188.1:389 \
  --user="uid=admin,cn=users,dc=yourserver,dc=com" \
  --password pw \
  --basedn "cn=users,dc=yourserver,dc=com" \
  --filter_user "uid=*" \
  --regex_groups "cn=(.*),cn=groups,dc=yourserver,dc=com" \
  --attrib_user uid \
  --attrib_groups memberOf \
```

```
  --attrib_password sambaNTPassword \
  --output /opt/fabinfra/bffh-data/config/users.toml

# restart bffhd
systemctl restart bffh.service
```

```
chmod +x /opt/fabinfra/scripts/fabaccess-users-toml-ldap/cron.sh
```

```
sudo vim /etc/cron.d/fabaccess-users-toml-ldap
```

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# At minute 0 past every hour.
0 */1 * * * bffh /opt/fabinfra/scripts/fabaccess-users-toml-ldap/cron.sh
```

Hinweise

Das Script basiert auf der Idee von https://gitlab.bht-berlin.de/innovisionlab/ldapbodge

Python Module

Das Script wurde getestet mit:

```
python -V
3.12.3

pip list installed
pip 24.0
pyasn1 0.6.1
pyasn1_modules 0.4.1
python-ldap 3.4.4
toml 0.10.2
```