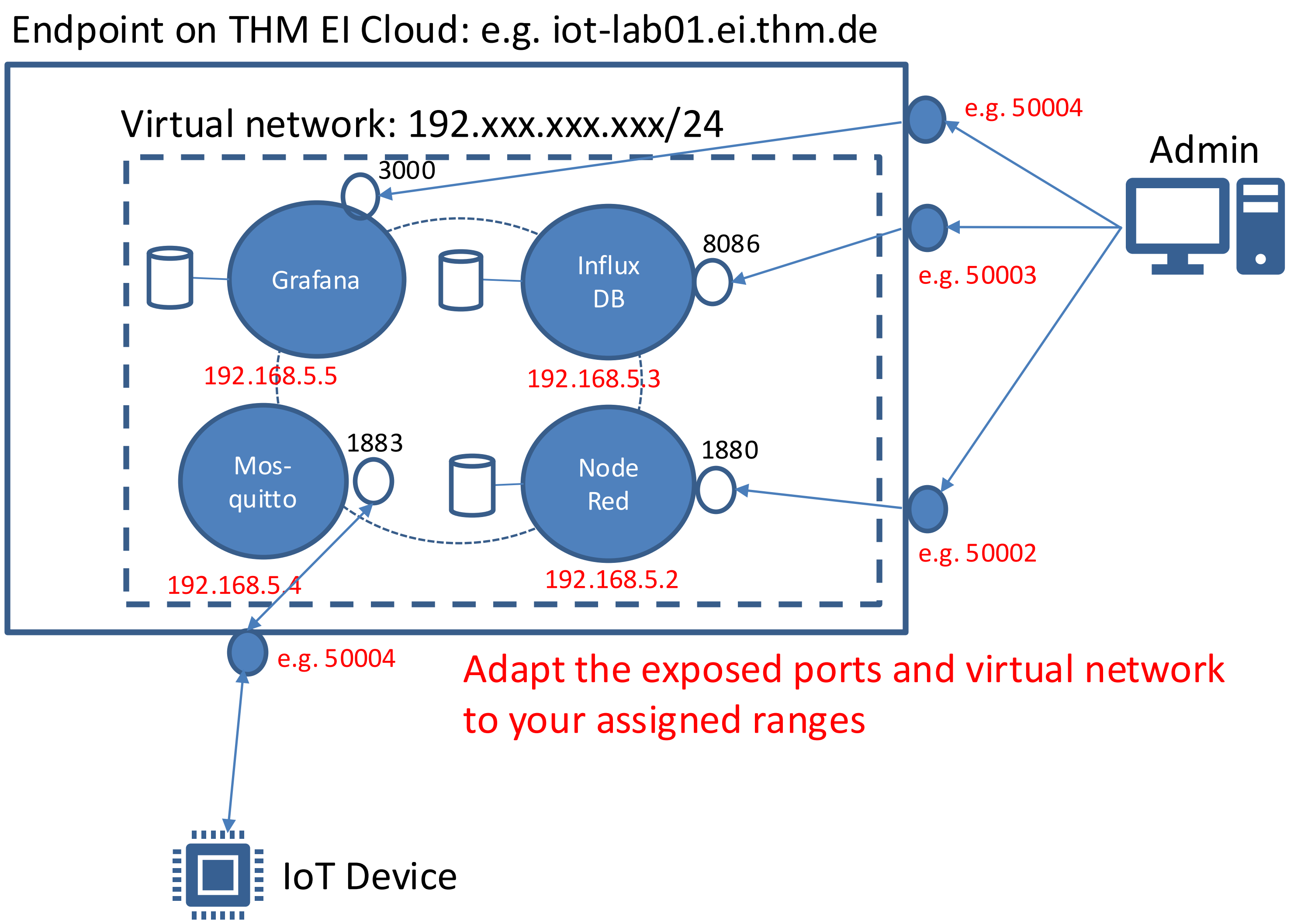


Adapt the exposed ports and virtual network to your assigned ranges

IoT Lab 2:

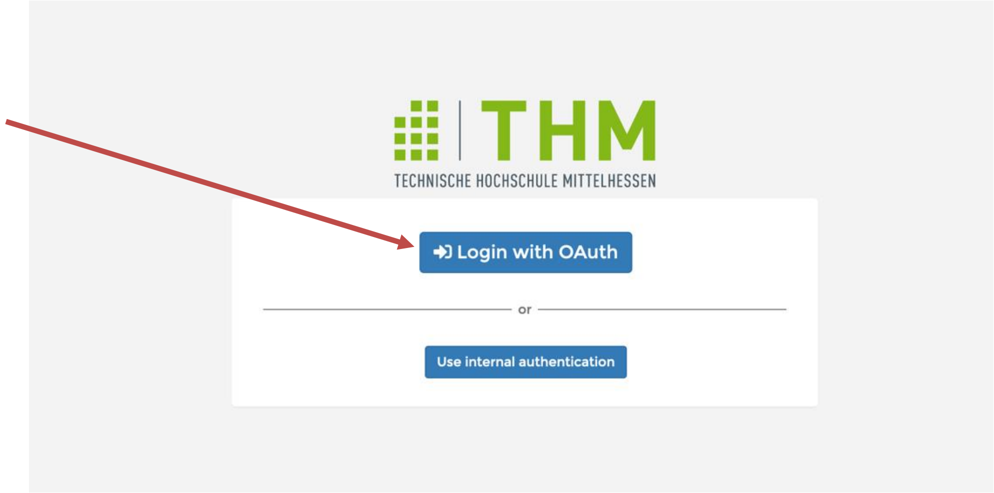
Deploying Docker Container Apps on the THM EI Cloud

Learning Objectives

- Using different cloud platform: EI IoT Cloud Platform (portainer)
- Deploy and configure container applications using a **YAML File** (Docker compose):
  - **NodeRed** with a local volume attached for persistent data + secure with password
  - **Mosquitto Broker** (no local volume is necessary)
  - **InfluxDB** with a local volume attached for persistent data
  - **Grafana** with a local volume attached for persistent data
- Use assigned **port-mapping**
- Create an easy „Sensordata“-Flow: publish local CPU load (from lab 1) via MQTT to own broker, subscribe in Node-Red in Cloud, store into database and visualize in grafana
- Subscribe to the defined topics in different ways using NodeRed, MQTT Explorer & MQTTx
- Learn how to secure your MQTT connection via TLS and how to use the secured EI MQTT Broker → used for future labs
- To deploy container there are mutiple options
  - a.) Portainer GUI (standalone or **yaml file**)
  - b.) Docker Command Line Interface ✅ ***(already done in Lab 1)***

Register and login to THM EI Cloud

- Navigate to https://cloud.ei.thm.de *
- Follow **Login with OAuth**
- Login with SSO with your THM credentials (like in Moodle, etc)
- Your should see an assigned environment, like IoT-Lab0x


THM
TECHNISCHE HOCHSCHULE MITTELHESSEN
Login with OAuth
or
Use internal authentication

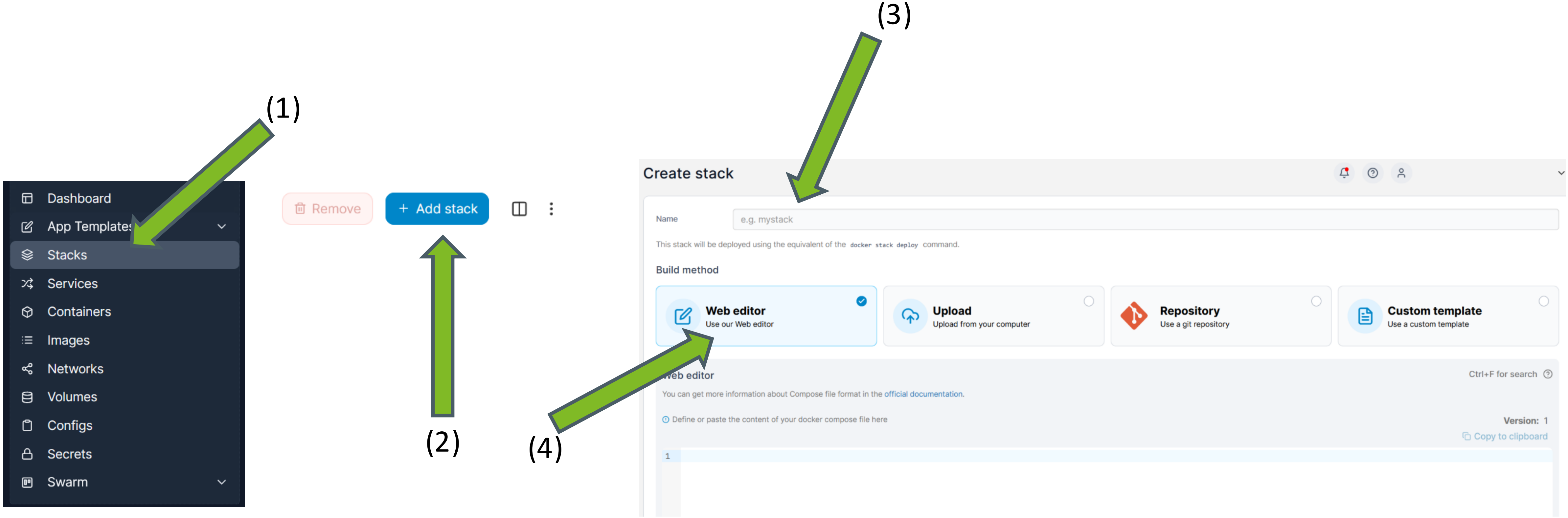
- * Note: If you are not in THM network,
  connect via VPN client
  https://www.thm.de/its/campusnetz/vpn.html

- Note down your port-range assigned to you in moodle / lecture → since you are all working on the same server, ports have to be unique per container!

  your port range: ______________________________

Create a new stack

- **Login** to your endpoint
  - Head to “Stacks” (1) and click “Add stack” (2)
  - Choose <yourname>_lab2_ws20xx as the name of your stack (3)
  - Use the web editor to configure your services (4) → see next slide



Create stack

Name: e.g. mystack

This stack will be deployed using the equivalent of the `docker stack deploy` command.

Build method

- ● Web editor – Use our Web editor
- ○ Upload – Upload from your computer
- ○ Repository – Use a git repository
- ○ Custom template – Use a custom template

Web editor

Ctrl+F for search

You can get more information about Compose file format in the official documentation.

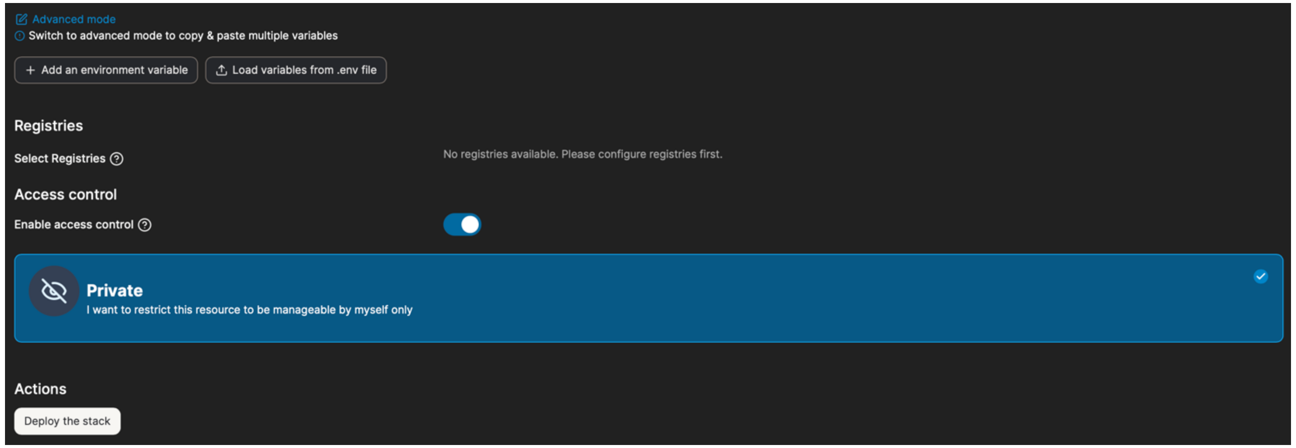
Define or paste the content of your docker compose file here

Version: 1

Copy to clipboard

Create a new stack

- **Deploy Stack using the docker-compose web editor:**
  - Add the services Mosquitto, NodeRed, InfluxDB and Grafana
  - Change <yourname> to your name (lowercase only)
  - Change the yxxxx ports to your own ports assigned by the tutor
  - Ensure that each service has its own unique port; do not reuse ports
  - Check your configuration and deploy the stack
    don’t get nervous, it takes about a minute to pull the images ☺
  - DO NOT copy from slides !!! → use the provided example
  - Attention: Indentation spaces are important



Advanced mode
Switch to advanced mode to copy & paste multiple variables

+ Add an environment variable
Load variables from .env file

Registries

Select Registries: No registries available. Please configure registries first.

Access control

Enable access control: ●

Private
I want to restrict this resource to be manageable by myself only ☑

Actions

Deploy the stack

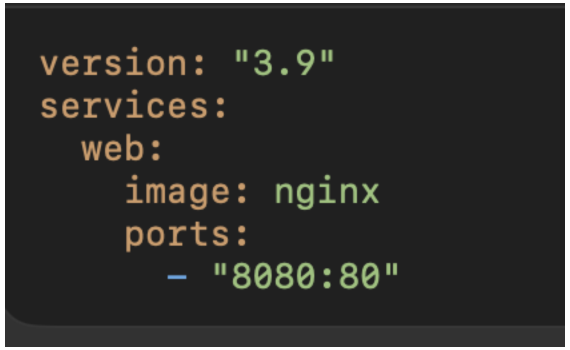
Docker-compose

```
version: "3.8"
services:
  node-red-<yourname>:
    image: nodered/node-red:latest
    ports:
      - "yxxxx:1880"
    networks:
      - <yourname>_net
    volumes:
      - <yourname>_nodered_vol:/data

  mosquitto-<yourname>:
    image: eclipse-mosquitto:1.6.13
    ports:
      - "yxxxx:1883"
    networks:
      - <yourname>_net

  influxdb-<yourname>:
    image: influxdb:latest
    ports:
      - "yxxxx:8086"
    networks:
      - <yourname>_net
    volumes:
      - <yourname>_influx_vol:/var/lib/influxdb2:rw

  grafana-<yourname>:
    image: grafana/grafana:latest
    ports:
      - "yxxxx:3000"
    networks:
      - <yourname>_net
    volumes:
      - <yourname>_grafana_vol:/var/lib/grafana:rw

volumes:
  <yourname>_nodered_vol:
  <yourname>_influx_vol:
  <yourname>_grafana_vol:

networks:
  <yourname>_net:
```

YAML File (some theory)

- **YAML** stands for **“YAML Ain’t Markup Language”** — it’s a simple, human-friendly format used to describe data. Think of it like a “recipe” that tells a system what to do.
- In the context of **Docker** or **Kubernetes**, YAML files are used to define:
  - Which containers to run
  - What settings they should have
  - How they should connect to each other (networking)
  - How many copies (scaling)
  - And more…
- **What happens behind the scenes?**
  - Once you write and run the YAML file:
  - Containers are **automatically deployed**
  - Docker (or Kubernetes) **manages** the services
  - **Scaling**, **networking**, **restart policies** — all are automated
- **So what do we need to define?**
- Just **what we want**, like:
  - The container image (e.g., nginx, node, mysql), Port mapping (e.g., 8080:80), Environment variables(opt), volumes, etc.

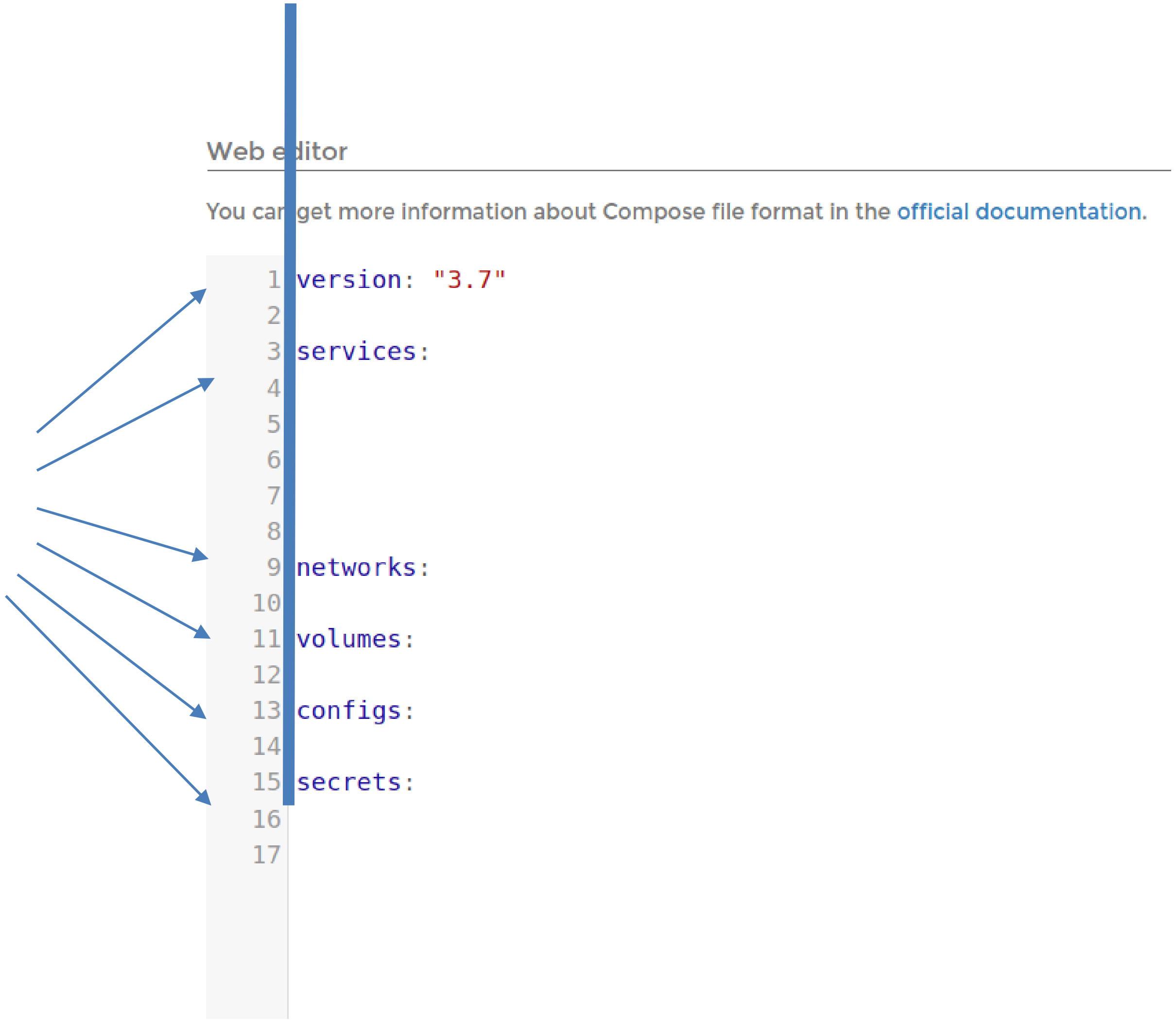
- This example tells Docker:
  - Run **nginx** web server (well-known port **80**)
  - Make it accessible from **port 8080** on our computer



```
version: "3.9"
services:
  web:
    image: nginx
    ports:
      - "8080:80"
```

YAML File Format

First level



Web editor

You can get more information about Compose file format in the official documentation.

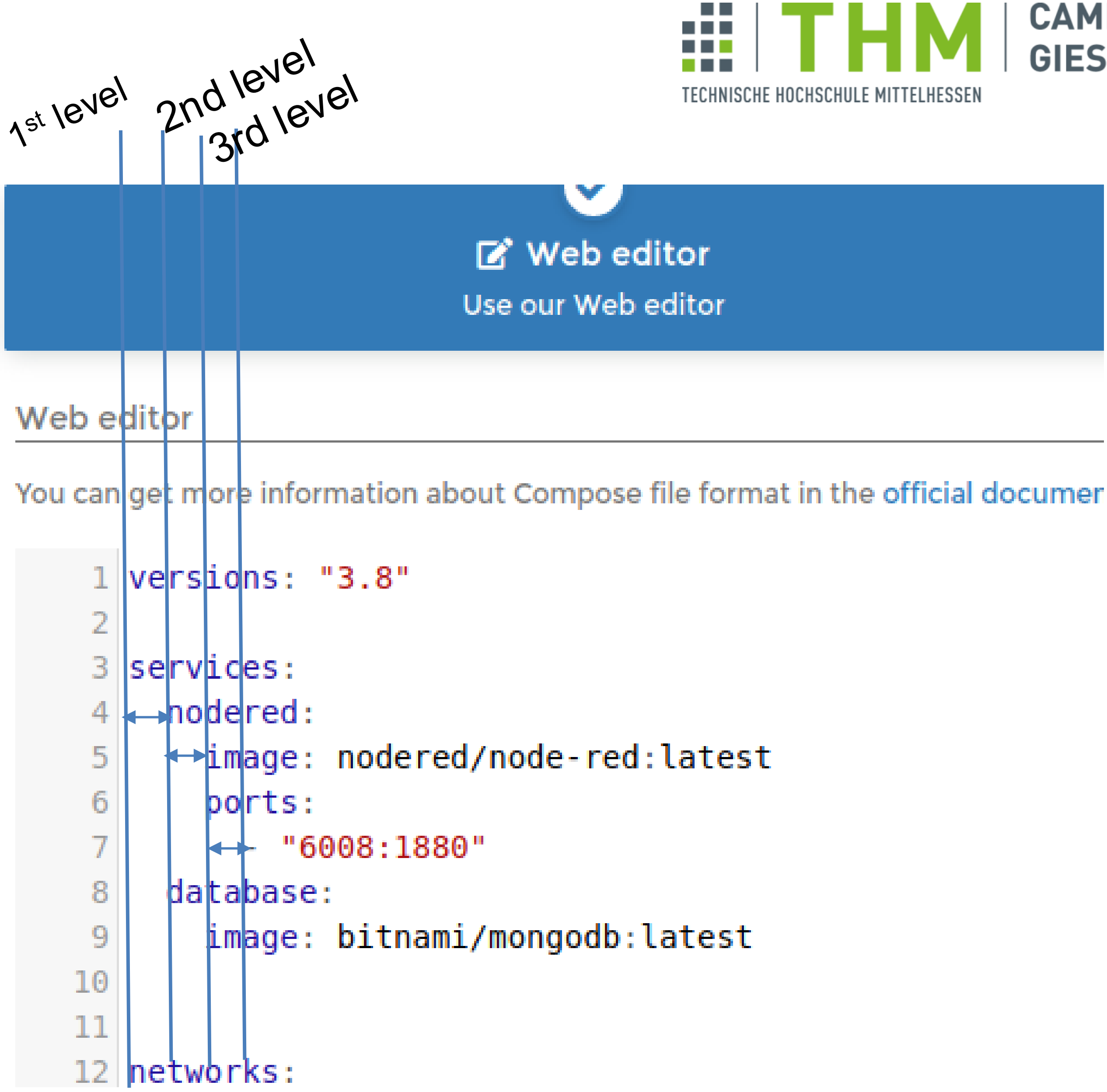
```
version: "3.7"

services:





networks:

volumes:

configs:

secrets:


```

Remember first level keywords

YAML File Format

How to write a YAML file:

- No Spaces
- No Upper Case Letters
- Use either Tab key or 2 spaces



1st level

2nd level

3rd level

Tab key or 2 spaces

Web editor

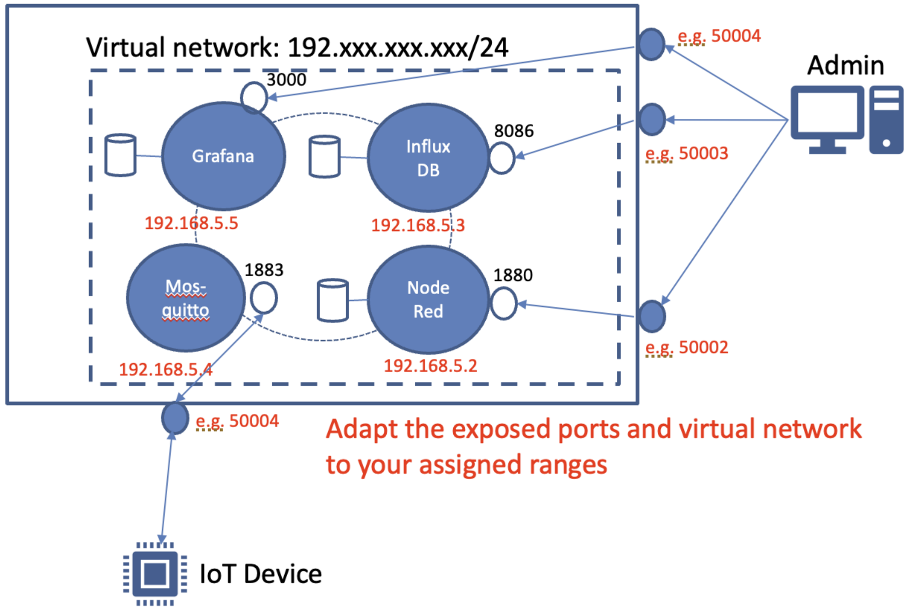
You can get more information about Compose file format in the official documer

```
versions: "3.8"

services:
  nodered:
    image: nodered/node-red:latest
    ports:
      - "6008:1880"
  database:
    image: bitnami/mongodb:latest


networks:
```

Access your applications

- Note down your IPs / Ports to access your applications:
- Node-Red
  Public IP: ________________ Public Port: ________________
  Private IP: ________________ Private Port: ________________
- Mosquitto
  Public IP: ________________ Public Port: ________________
  Private IP: ________________ Private Port: ________________
- InfluxDB
  Public IP: ________________ Public Port: ________________
  Private IP: ________________ Private Port: ________________
- Grafana
  Public IP: ________________ Public Port: ________________
  Private IP: ________________ Private Port: ________________

Endpoint on THM EI Cloud: e.g. iot-lab01.ei.thm.de



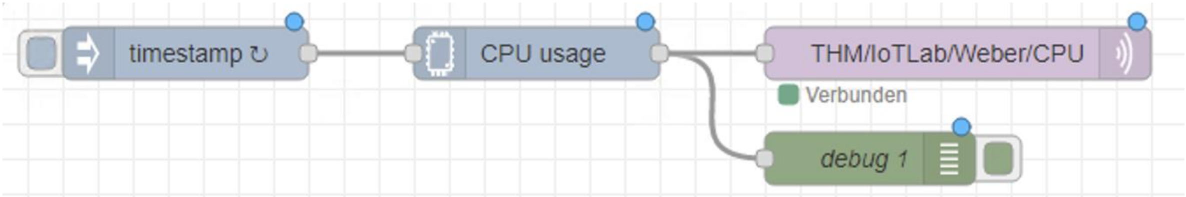
Adapt the exposed ports and virtual network to your assigned ranges

Configure Node Red & MQTT

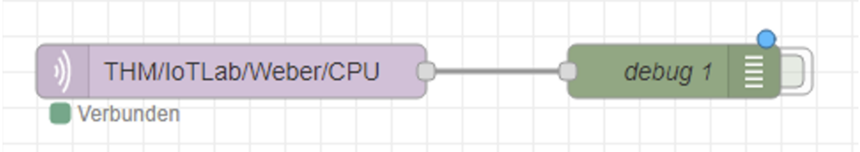
- Open **NodeRed on your local computer (from lab 1)**
  - Publish the CPU Data to either your broker (a) or EI Broker (b)
    (a) use your own broker via

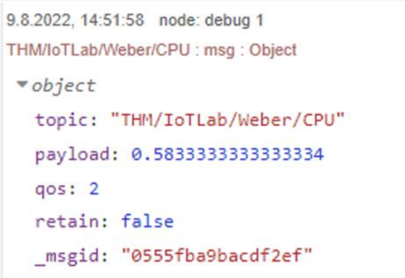
    Public IP: _______________ Public Port: _________________
    Private IP: ________________ Private Port: _____________________

    (b) Use EI Broker: mqtt.ei.thm.de:1993 user/pw: **iotlab/iotlab** Topic format: ***THM/IoTLab/yourname***



- Open **NodeRed deployed on cloud (this lab)**
  - subscribe to your published data and show values in debug

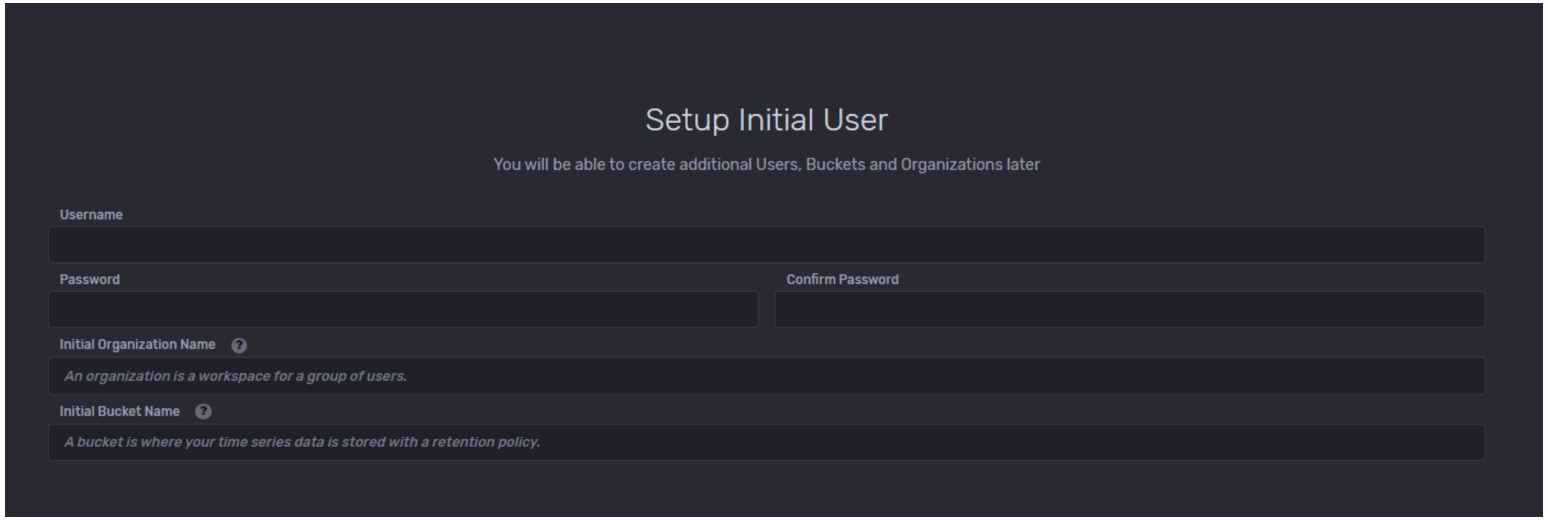




```
9.8.2022, 14:51:58   node: debug 1
THM/IoTLab/Weber/CPU : msg : Object
▾object
  topic: "THM/IoTLab/Weber/CPU"
  payload: 0.5833333333333334
  qos: 2
  retain: false
  _msgid: "0555fba9bacdf2ef"
```

Configure InfluxDB

- **Open InfluxDB**
  - Walk through the Get Started guide and setup initial user
    - Choose a username and password and **note it down**
    - Choose the Organization Name: THM
    - Name your data bucket: lab02



Setup Initial User

You will be able to create additional Users, Buckets and Organizations later

Username

Password

Confirm Password

Initial Organization Name

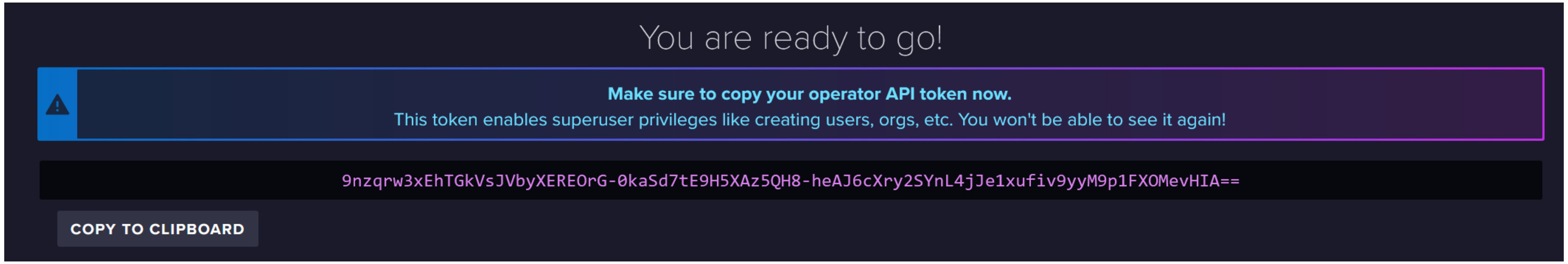
An organization is a workspace for a group of users.

Initial Bucket Name

A bucket is where your time series data is stored with a retention policy.

Configure InfluxDB

- **Save your InfluxDB API key (use e.g. editor)**
  - Make a note of the API key, as we'll be using it later.
  - → COPY TO CLIPBOARD not always working, so copy manually !!



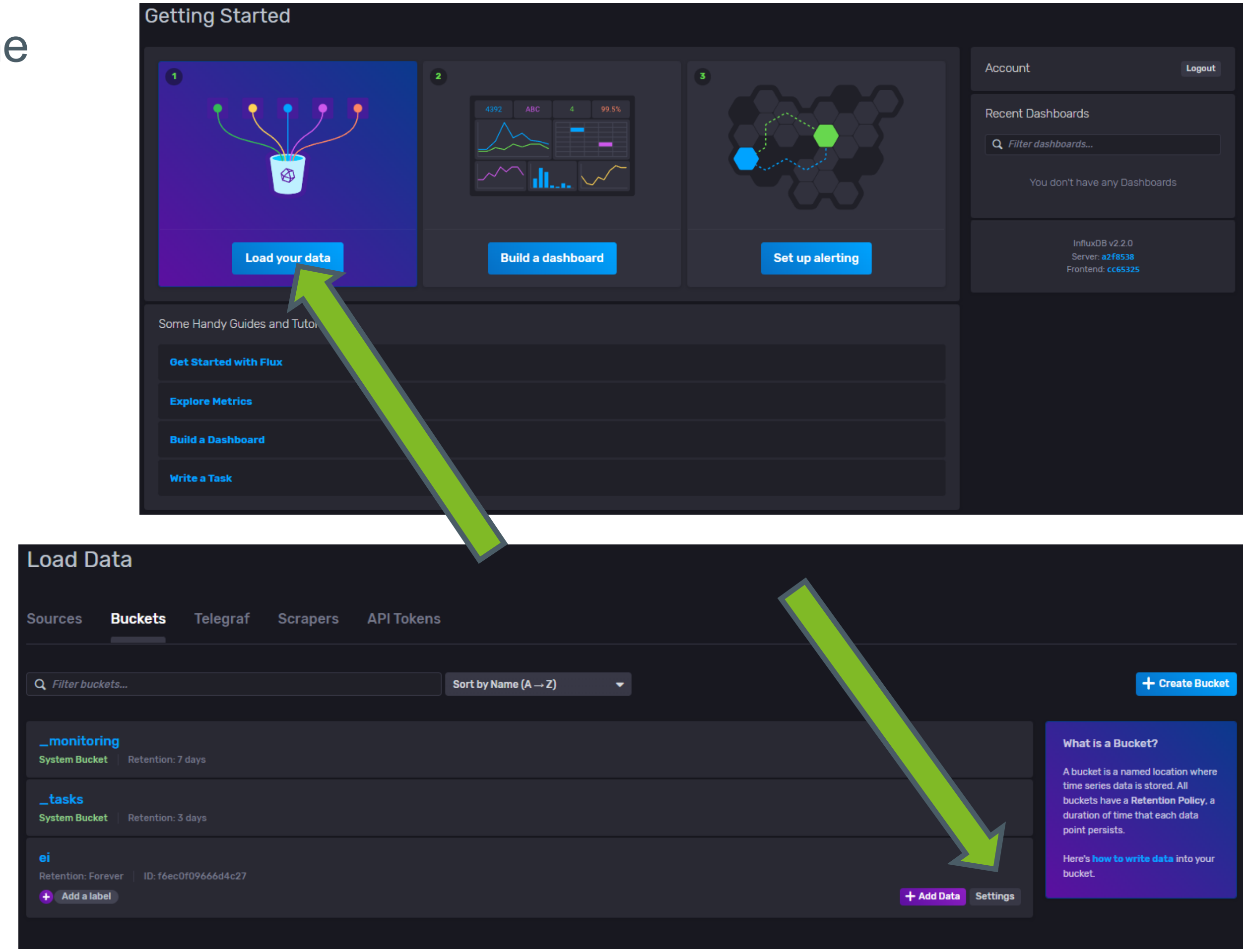
You are ready to go!

Make sure to copy your operator API token now.
This token enables superuser privileges like creating users, orgs, etc. You won't be able to see it again!

```
9nzqrw3xEhTGkVsJVbyXEREOrG-0kaSd7tE9H5XAz5QH8-heAJ6cXry2SYnL4jJe1xufiv9yyM9p1FXOMevHIA==
```

Configure InfluxDB

- **Configure InfluxDB**
  - Click “Load your data” on the Getting Started Page
  - In the “Data” menu, head to Buckets and change the retention policy in “settings” if needed



Getting Started

1 Load your data

2 Build a dashboard

3 Set up alerting

Account Logout

Recent Dashboards

Filter dashboards...

You don't have any Dashboards

InfluxDB v2.2.0
Server: a2f8538
Frontend: cc65325

Some Handy Guides and Tutori

- Get Started with Flux
- Explore Metrics
- Build a Dashboard
- Write a Task

Load Data

Sources | Buckets | Telegraf | Scrapers | API Tokens

Filter buckets... | Sort by Name (A → Z) | + Create Bucket

_monitoring
System Bucket | Retention: 7 days

_tasks
System Bucket | Retention: 3 days

ei
Retention: Forever | ID: f6ec0f09666d4c27
+ Add a label | + Add Data | Settings

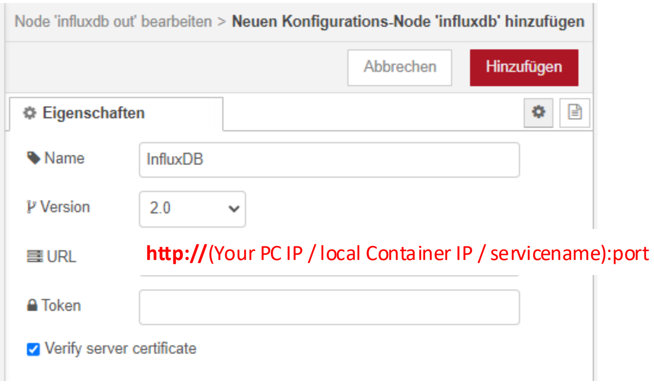
What is a Bucket?

A bucket is a named location where time series data is stored. All buckets have a **Retention Policy**, a duration of time that each data point persists.

Here's how to write data into your bucket.

Configure InfluxDB in NodeRed

- **Add InfluxDB Server to Node Red (cloud)**
  - Go back to your Node Red Service and add an influxdb out node
    - needs to be installed → manage palette
  - Configure a new InfluxDB Server and select version 2
  - Enter your URL (**http://** _+ IP/FQDN cloud endpoint or local Container IP or service name_ + **port** of InfluxDB) and the copied admin token
  - Add the configuration
  - Edit the influxdb out node and enter your organization, bucket, measurement
    - Organization: THM
    - Bucket: lab02
    - Measurement: CPU_PC


Node 'influxdb out' bearbeiten > Neuen Konfigurations-Node 'influxdb' hinzufügen
Abbrechen
Hinzufügen
Eigenschaften

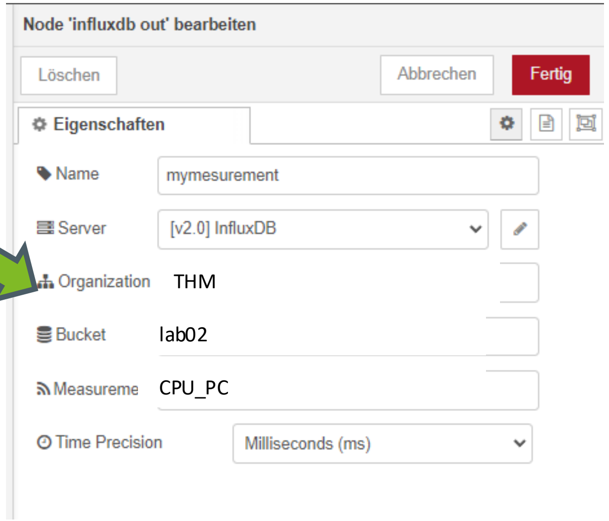
Name: InfluxDB

Version: 2.0

URL: **http://**(Your PC IP / local Container IP / servicename):port

Token:

☑ Verify server certificate



Name: mymesurement

Server: [v2.0] InfluxDB

Organization: THM

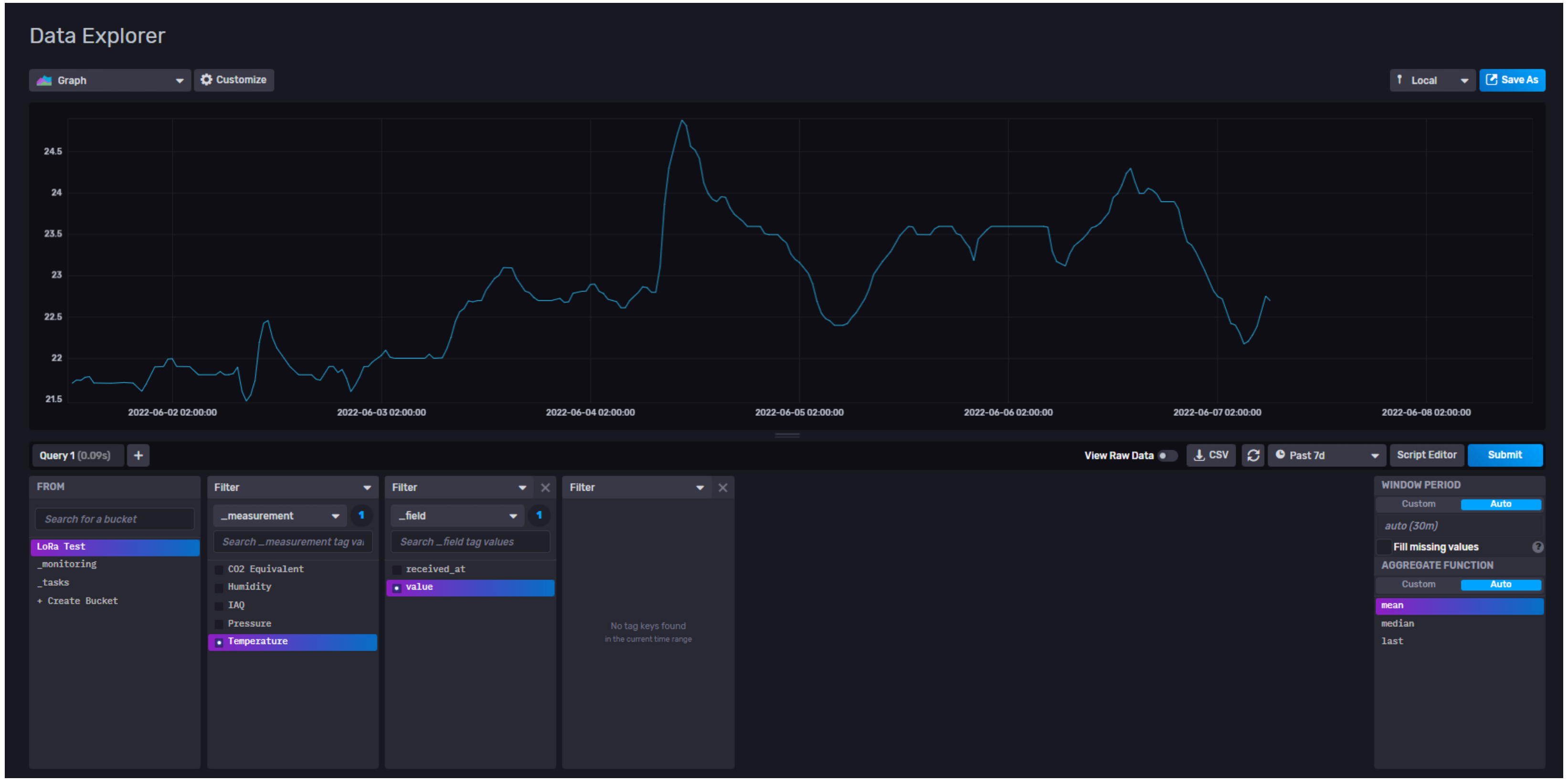
Bucket: lab02

Measureme: CPU_PC

Time Precision: Milliseconds (ms)

Configure InfluxDB

- **View your measurements**
  - On your InfluxDB Web-UI go to the Explore menu
  - You can now select your bucket and view your measurements and click submit



Data Explorer

Query 1 (0.09s)

FROM
Search for a bucket
LoRa Test
_monitoring
_tasks
+ Create Bucket

Filter
_measurement 1
Search _measurement tag va
□ CO2 Equivalent
□ Humidity
□ IAQ
□ Pressure
● Temperature

Filter
_field 1
Search _field tag values
□ received_at
● value

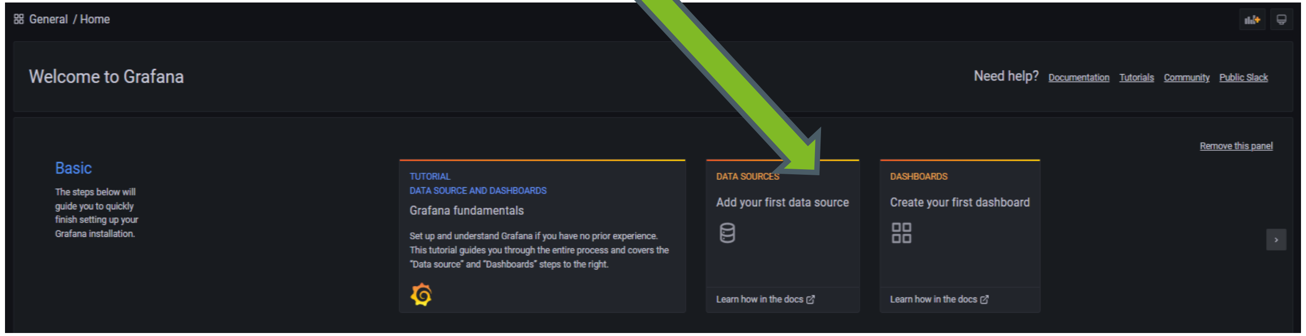
Filter
No tag keys found
in the current time range

WINDOW PERIOD
Custom | Auto
auto (30m)
□ Fill missing values

AGGREGATE FUNCTION
Custom | Auto
mean
median
last

Configure Grafana

- **Open the Grafana Web Interface**
  - Login with the default user/password combination (admin/admin)
  - Choose a new password and **note it down**
  - On the Grafana Homepage click the “Add your first data source” Panel
  - Add InfluxDB as a source



General / Home

Welcome to Grafana

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Basic

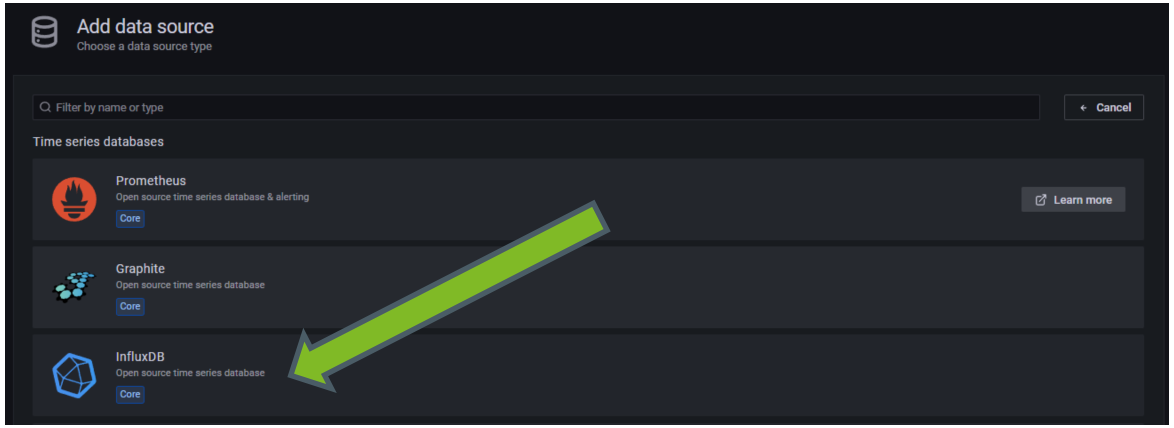
The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the “Data source” and “Dashboards” steps to the right.

DATA SOURCES
Add your first data source
Learn how in the docs

DASHBOARDS
Create your first dashboard
Learn how in the docs



Add data source
Choose a data source type

Filter by name or type

← Cancel

Time series databases

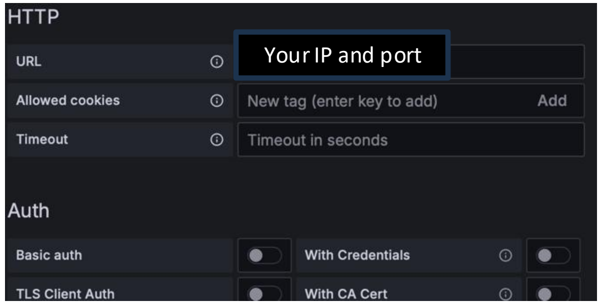
Prometheus
Open source time series database & alerting
Core
Learn more

Graphite
Open source time series database
Core

InfluxDB
Open source time series database
Core

Configure Grafana

- **Add InfluxDB to Grafana**
  - Change the Query Language to **Flux**
  - Enter the URL and Port of your InfluxDB
  - Disable Basic Auth
  - Enter your **Organization**, **Token** and **Default Bucket** under "InfluxDB Details"
  - Click save and test → should be green and successful

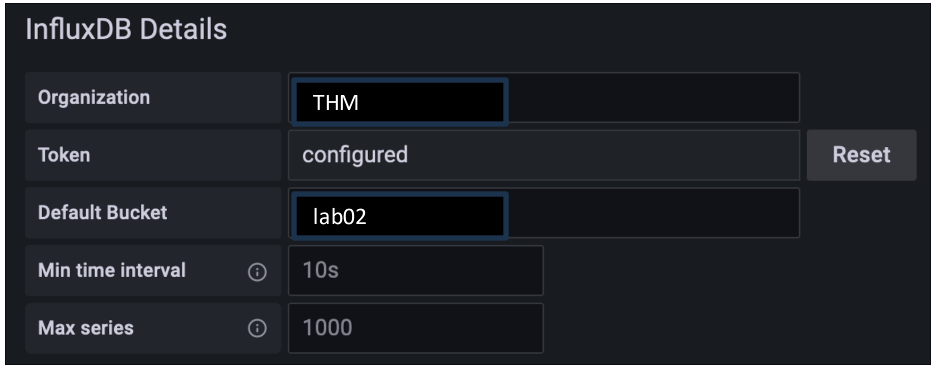


HTTP

URL	Your IP and port
Allowed cookies	New tag (enter key to add) Add
Timeout	Timeout in seconds

Auth

Basic auth	○	With Credentials	○
TLS Client Auth	○	With CA Cert	○

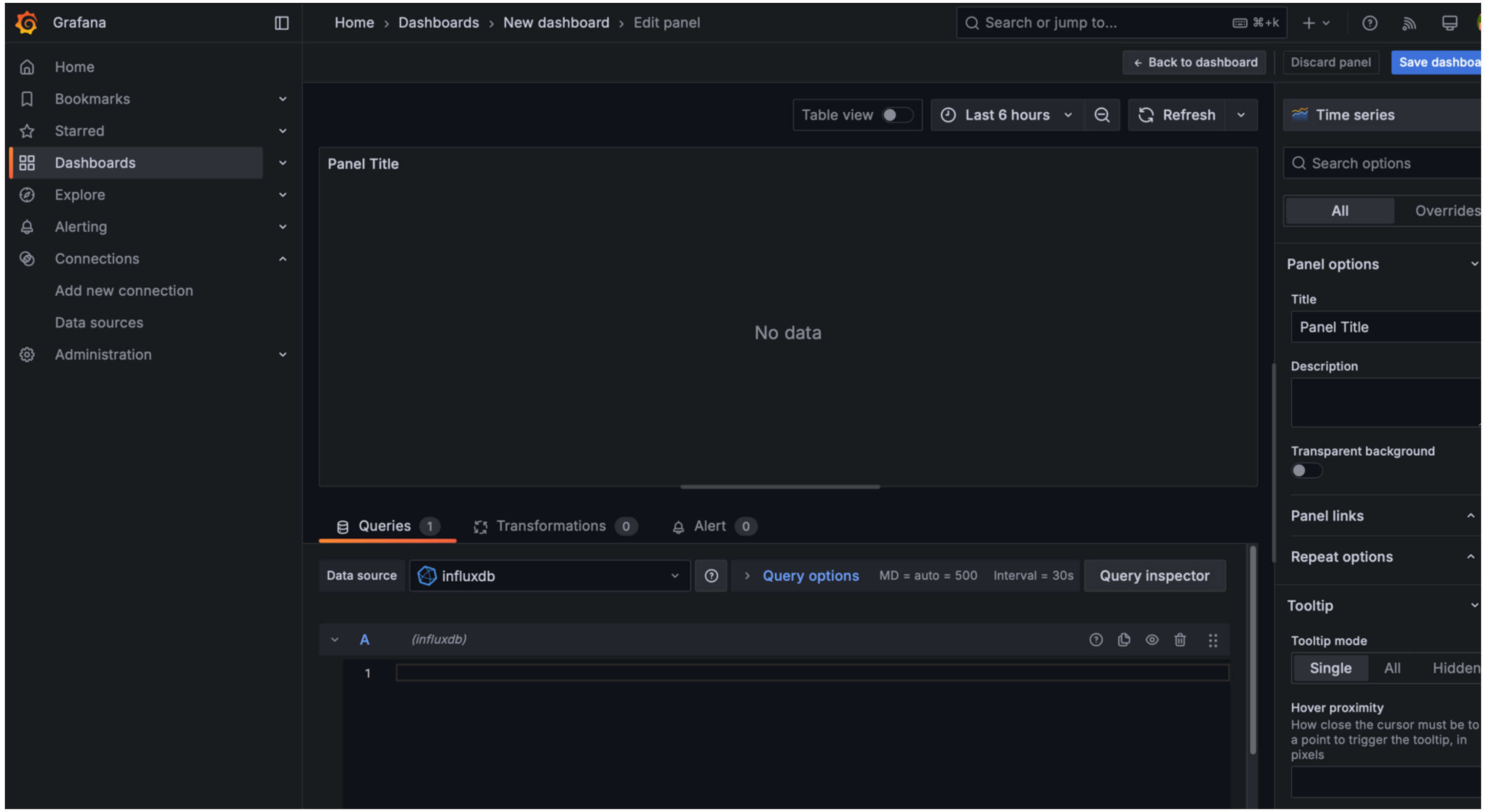


InfluxDB Details

Organization	THM	
Token	configured	Reset
Default Bucket	lab02	
Min time interval	10s	
Max series	1000	

Configure Grafana

- **Create a dashboard and display measurements**
  - Click on the “Create your first dashboard” panel at the Homepage of Grafana
  - Add visualization and select your influxdb as data source



Home › Dashboards › New dashboard › Edit panel

Data source: influxdb

Query options MD = auto = 500 Interval = 30s

A (influxdb)

Title: Panel Title

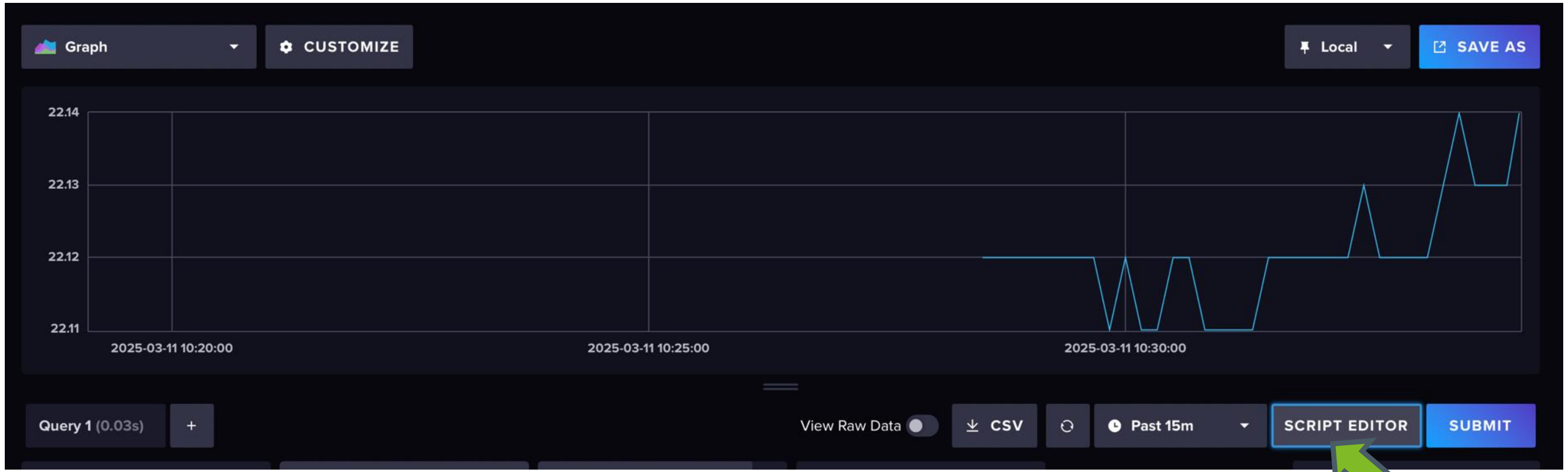
Transparent background

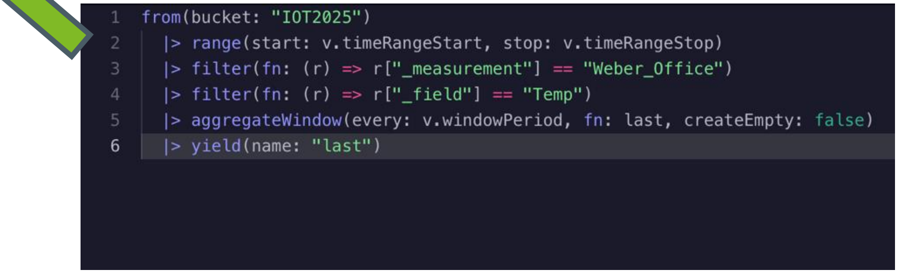
Tooltip mode: Single

Hover proximity
How close the cursor must be to a point to trigger the tooltip, in pixels

Configure Grafana

- **Go back to your InfluxDB Explorer**
  - Choose your bucket and a measurement you want to visualize (value)
  - Click on the script editor button
  - Copy the query text for Grafana

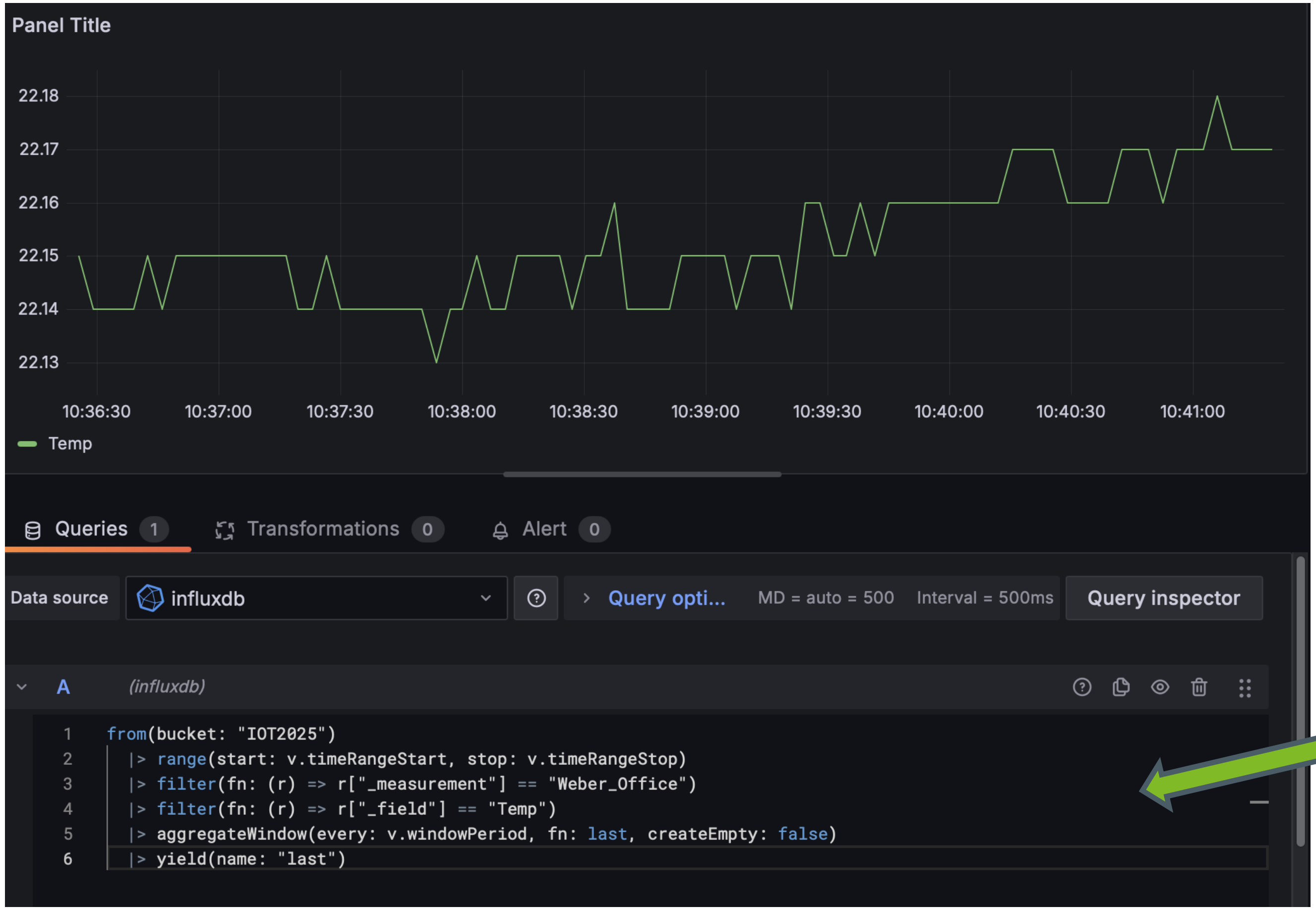




```
from(bucket: "IOT2025")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "Weber_Office")
  |> filter(fn: (r) => r["_field"] == "Temp")
  |> aggregateWindow(every: v.windowPeriod, fn: last, createEmpty: false)
  |> yield(name: "last")
```

Configure Grafana

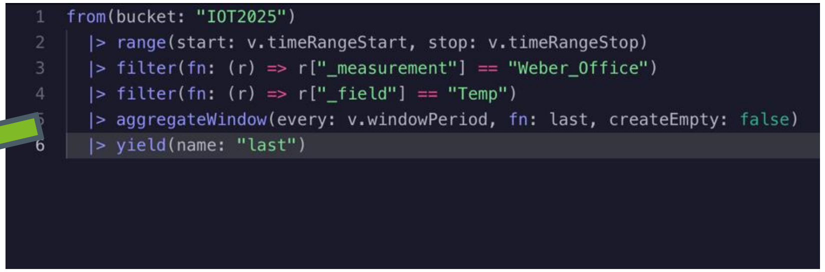
- Insert the copied Query text in your Grafana raw query editor



Data source: influxdb

MD = auto = 500 Interval = 500ms

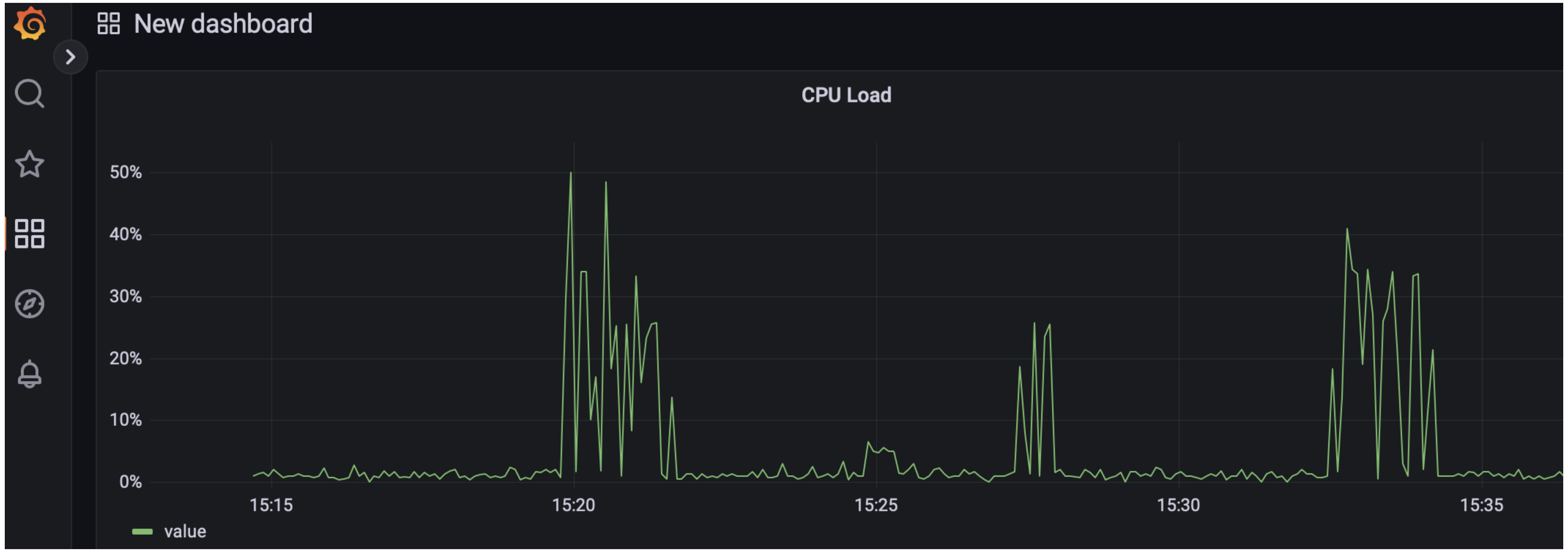
```
from(bucket: "IOT2025")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "Weber_Office")
  |> filter(fn: (r) => r["_field"] == "Temp")
  |> aggregateWindow(every: v.windowPeriod, fn: last, createEmpty: false)
  |> yield(name: "last")
```

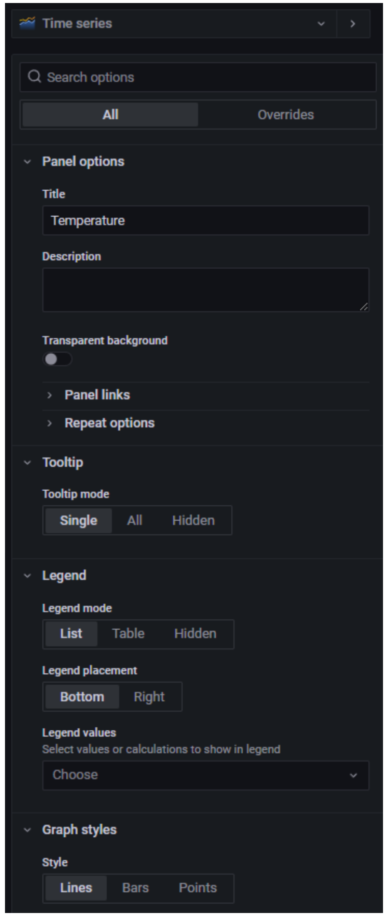


```
from(bucket: "IOT2025")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "Weber_Office")
  |> filter(fn: (r) => r["_field"] == "Temp")
  |> aggregateWindow(every: v.windowPeriod, fn: last, createEmpty: false)
  |> yield(name: "last")
```

Configure Grafana

- **Now your turn:**
  - **Give a title, description and the correct unit**
    - Fine tune your graph if you want to
    - Click apply if you are done and save your dashboard
      It will not be saved automatically!





Title: Temperature

Description:

Transparent background

Tooltip mode: [✓ Single] All Hidden

Legend mode: [✓ List] Table Hidden

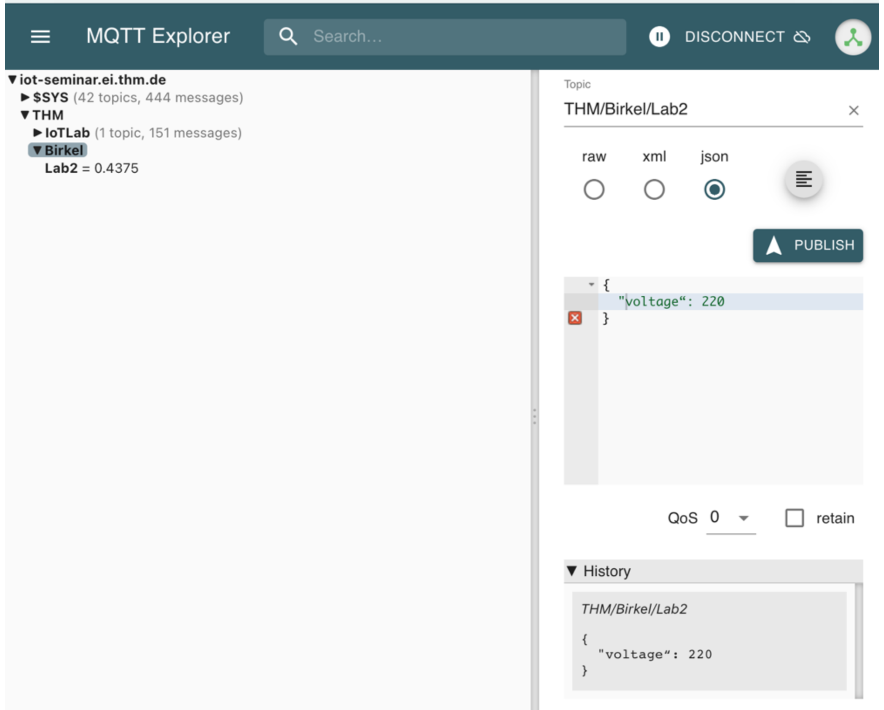
Legend placement: [✓ Bottom] Right

Legend values
Select values or calculations to show in legend
Choose

Style: [✓ Lines] Bars Points

Experiment with MQTT Clients

- **MQTT Explorer**
- **Open MQTT Explorer (already installed in Lab)**
    - Add a new connection Server: either your own broker or EI broker
    - Subscribe to your topic and show your data



```
▼iot-seminar.ei.thm.de
  ▶$SYS (42 topics, 444 messages)
  ▼THM
    ▶IoTLab (1 topic, 151 messages)
    ▼Birkel
      Lab2 = 0.4375
```

Topic: THM/Birkel/Lab2

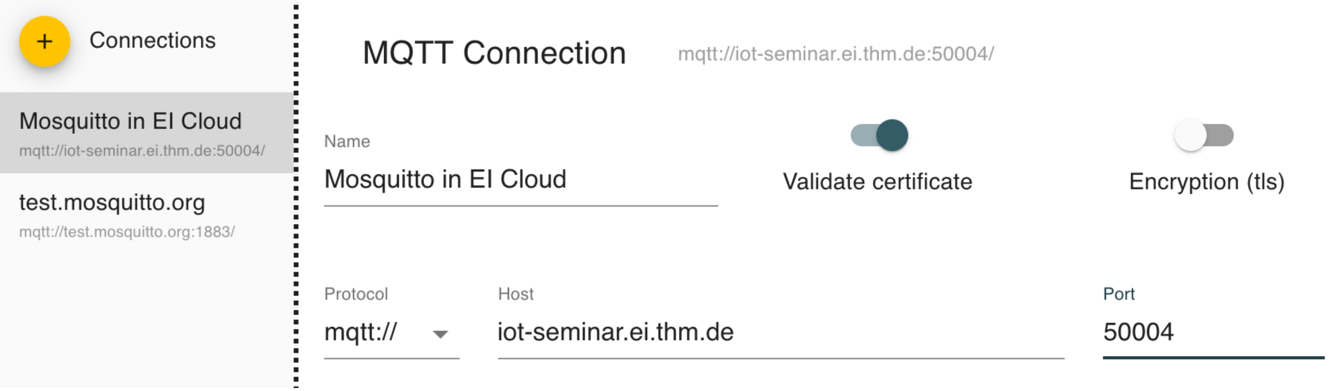
○ raw ○ xml ● json

```
{
  "voltage“: 220
}
```

QoS 0 □ retain

THM/Birkel/Lab2

```
{
  "voltage“: 220
}
```



Mosquitto in EI Cloud
mqtt://iot-seminar.ei.thm.de:50004/

test.mosquitto.org
mqtt://test.mosquitto.org:1883/

MQTT Connection mqtt://iot-seminar.ei.thm.de:50004/

Name: Mosquitto in EI Cloud

Validate certificate: on

Encryption (tls): off

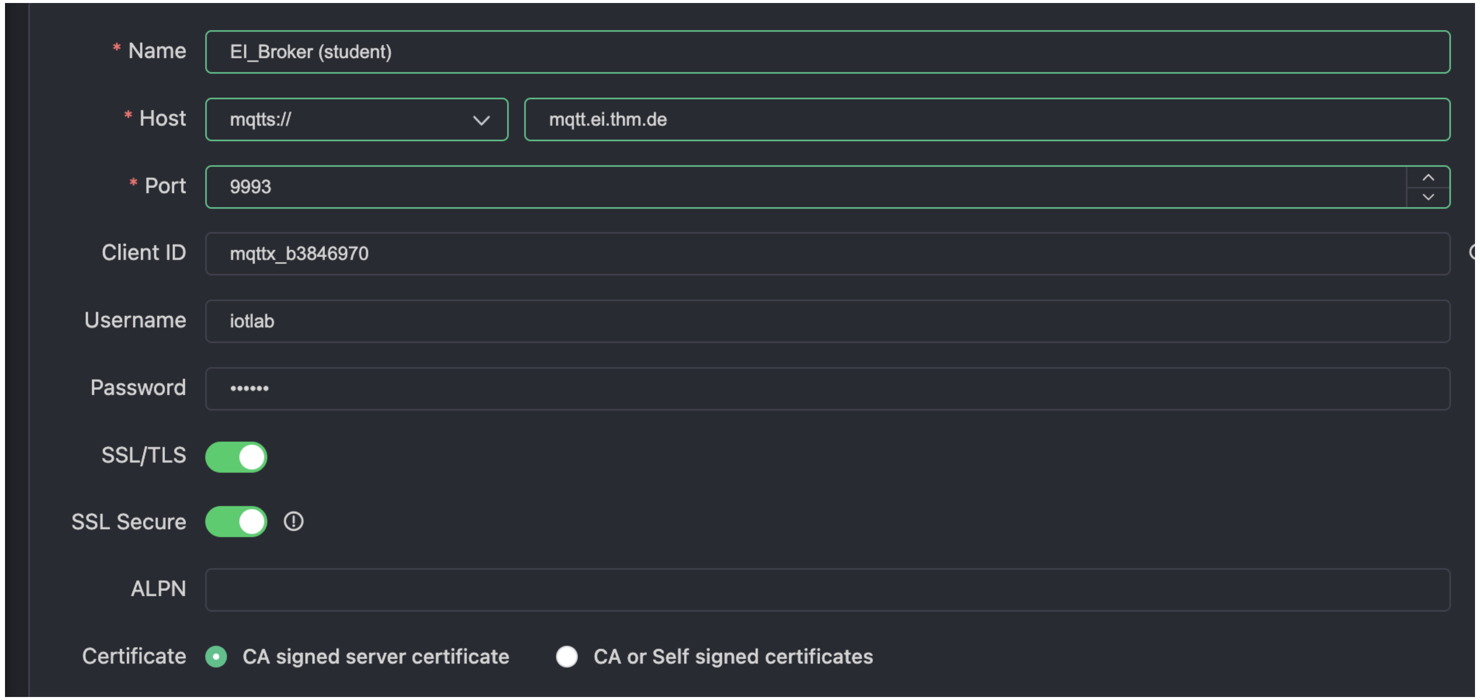
Protocol: mqtt://

Host: iot-seminar.ei.thm.de

Port: 50004

Experiment with MQTT Clients

- **MQTTx**
- **Open MQTTx (now we use TLS on EI Broker)**
  - Add a new connection Server: either your own broker or EI broker
  - Subscribe to your topic and show your data



* Name: EI_Broker (student)

* Host: mqtts:// mqtt.ei.thm.de

* Port: 9993

Client ID: mqttx_b3846970

Username: iotlab

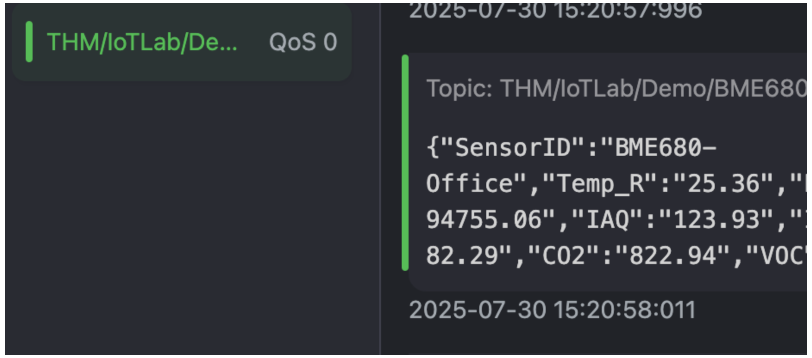
Password: ••••••

SSL/TLS: on

SSL Secure: on

ALPN:

Certificate: ● CA signed server certificate ○ CA or Self signed certificates



THM/IoTLab/De... QoS 0

2025-07-30 15:20:57:996

Topic: THM/IoTLab/Demo/BME680

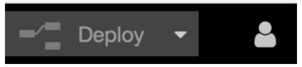
```
{"SensorID":"BME680-
Office","Temp_R":"25.36","
94755.06","IAQ":"123.93","
82.29","CO2":"822.94","VOC
```

2025-07-30 15:20:58:011

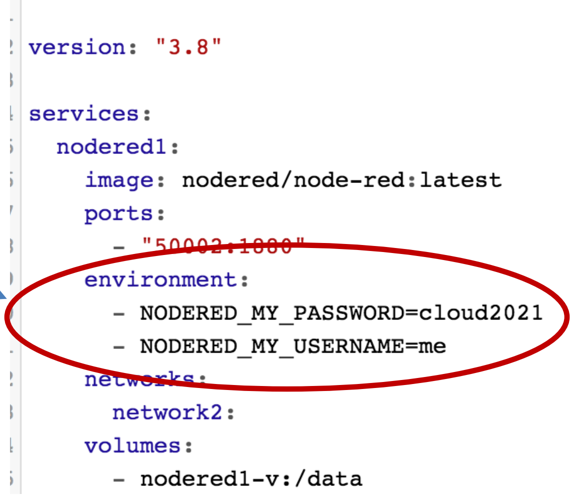
Add Authentication to NodeRed

- ***Option 1*: Modify setting.js file to define username&password**
- Go to your NodeRed volume and download setting.js, go to // securing node red section in this file
- Uncomment this string including the users key value which has one item defined as a JSON Object
  - JSON Objects defines: Username, password and permissions
  - We can generate the hash of the password via a hashfunction:
    `password: require('bcryptjs‘).hashSync(“cloud2021“)`
    Libary bccrpyt includes the hashSync function which returns the hash value of the password
  - Rename old setting.js to setting.js.backup and upload modified file including the password
  - Go to your stack, choose NodeRed and update it => NodeRed will be restartet and reads the settings.js file
  - Login with your username and password
  - **Don‘t forget to logout in NodeRed on the upper right corner!**

```
// Securing Node-RED
// ----------------
adminAuth: {
   type: "credentials",
   users: [{
      username: "admin",
      password: require('bcryptjs').hashSync("cloud2021"),
       permissions: "*“
   }]
 },
```



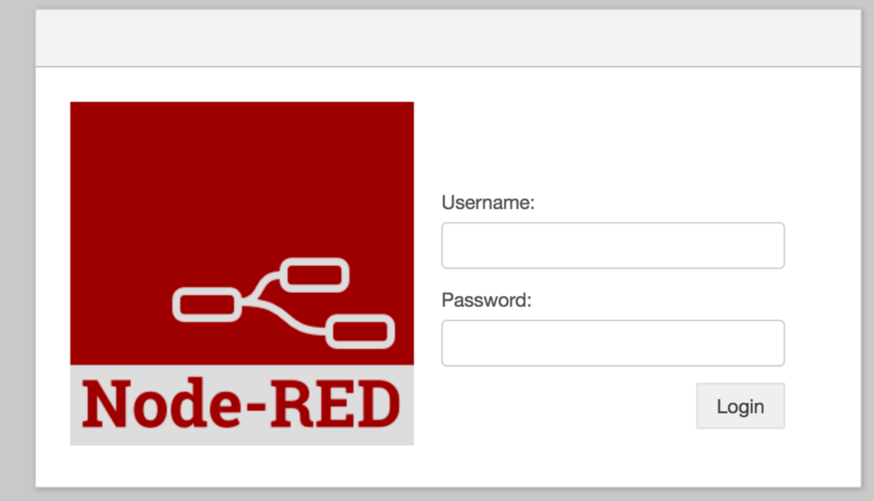
- **Not used now: Option 2: Using an environment Variables to define username & pw**
  - Environment variables exist at the runtime of the container
  - Modify your deployment stack file (YAML) and define username and password
  - Update your settings.js file to read the new environment variable:
    - Go to your nodered volume
    - Download settings.js file
    - Rename the original setting.js file into setting.js.backup
    - Modify it according to the slide on the next page & upload it again
    - Restart the NodeRed app and login with the username and password to node red
    - Modify the password in the deployment file and test if it works!



```
version: "3.8"

services:
  nodered1:
    image: nodered/node-red:latest
    ports:
      - "50002:1880"
    environment:
      - NODERED_MY_PASSWORD=cloud2021
      - NODERED_MY_USERNAME=me
    networks:
      network2:
    volumes:
      - nodered1-v:/data
```

Modifying the file settings.js on volume

```
//All the way on the to in settings.js enter the following function:
my_function = ()=> {

    username = process.env.NODERED_MY_USERNAME
    password = process.env.NODERED_MY_PASSWORD

    return {
        username,
        password
    }
}

// Securing Node-RED
    // -----------------
    // To password protect the Node-RED editor and admin API, the following
    // property can be used. See http://nodered.org/docs/security.html for details.
    adminAuth: {
        type: "credentials",
        users: [{
            username: my_function()['username'],
            password: require('bcryptjs').hashSync(my_function()['password']),
            permissions: "*"
        }]
    },
```



Lab Summary

EI IoT Cloud platform, Docker compose, secure IoT services and MQTTs

In this lab, you learned to:

- **Deploy IoT Applications in the Cloud 🐳**
  - Use Portainer GUI and Docker Compose YAML for container deployment
  - Manage Node-RED, Mosquitto, InfluxDB, and Grafana with volume and port configuration + securing NodeRed with username and password
- **Build and Visualize IoT Data Flows 🔄**
  - Publish system metrics via MQTT
  - Subscribe in Node-RED (cloud)
  - Store in InfluxDB
  - Visualize using Grafana dashboards
- **Work Securely with MQTT 🔐**
  - Subscribe using multiple clients (Node-RED, MQTT Explorer, MQTTx)
  - Secure connections with TLS
  - Use the **EI MQTT Broker** with certificates

🎯 You now understand

- **multi-container deployment**
- **cloud workflows**
- **MQTT and service security**
- **→ all essential for scalable IoT solutions.**

Lab 2 Review & Reflection

- What is the purpose of Docker Compose?
- Which services did you deploy using a YAML file?
- Why is port mapping necessary in container-based applications?
- How do you persist data for containers like Node-RED, InfluxDB, and Grafana?
- What is the purpose of an MQTT Broker? Which are the well-known ports?

Lab 2 Review & Reflection

- What is TLS and why is it important for MQTT communication?

- Which elements are used to secure connection to the used MQTT Broker?

- Create a topology for this lab (template is provided)