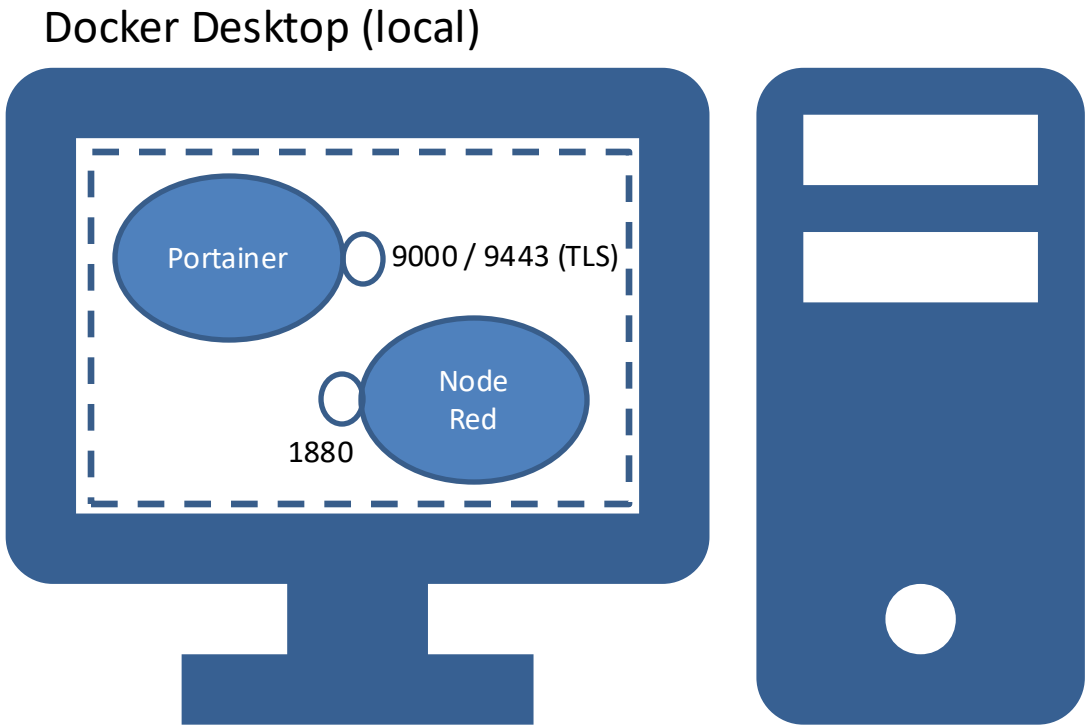


IoT Seminar Lab 1:

Local Docker „Cloud“

Docker Desktop - Getting started

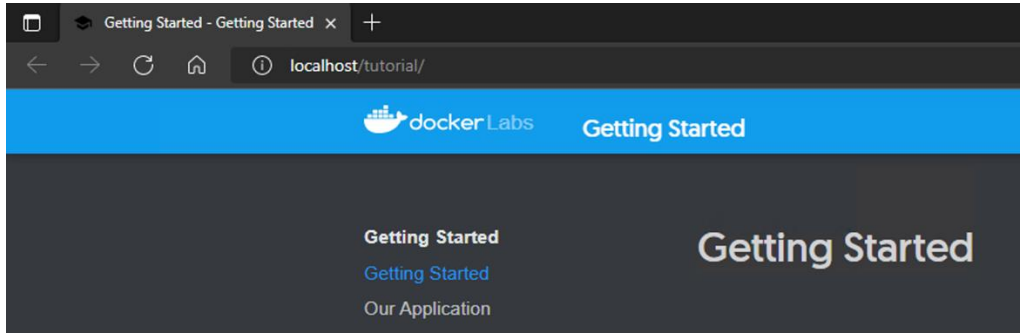
THM CAMPUS GIESSEN | EI Elektro- und Informationstechnik
TECHNISCHE HOCHSCHULE MITTELHESSEN

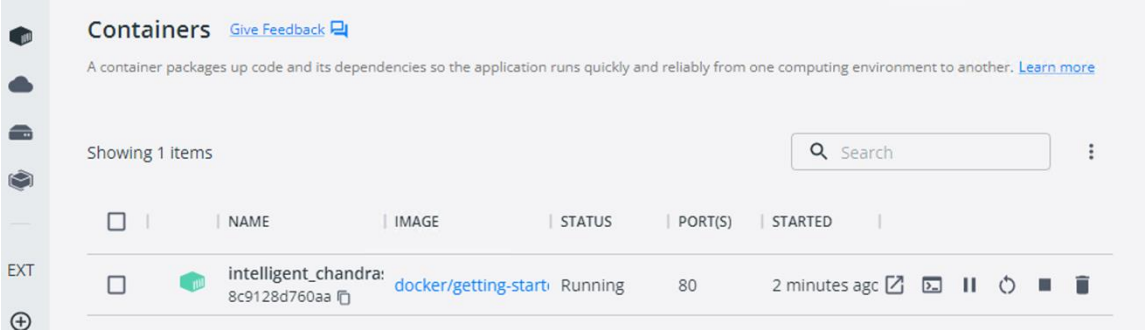
- Run once the following file: C:/zIT/zDocker before starting docker desktop, otherwise restart PC and run script first
- Start your local Docker Desktop and open cmd / terminal
  - Check with command *docker version* if docker is running correctly (see lab 0)
- If you like, start tutorial (@home)
- First we run a Sample Container as shown in (cmd)
  - *docker run -d -p 80:80 docker/getting-started*
- In Docker Desktop GUI you should see a running container now
- Open a browser with localhost or
  click on button “open with browser” at your container

HINT: this script deletes ALL your docker data, so do not run it during semester if not needed !!!

```
U:\>C:

C:\>docker run -d -p 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
df9b9388f04a: Pull complete
5867cba5fcbd: Pull complete
4b639e65cb3b: Pull complete
061ed9e2b976: Pull complete
bc19f3e8eeb1: Pull complete
4071be97c256: Pull complete
79b586f1a54b: Pull complete
0c9732f525d6: Pull complete
Digest: sha256:b558be874169471bd4e65bd6eac8c303b271a7ee8553ba47481b73b2bf597aae
Status: Downloaded newer image for docker/getting-started:latest
8c9128d760aa2afe53bd6da4e8de51f6ca1e603c8a4948be27ca4b404b4c0d36
```





- Congratulations: your first container is working ☺

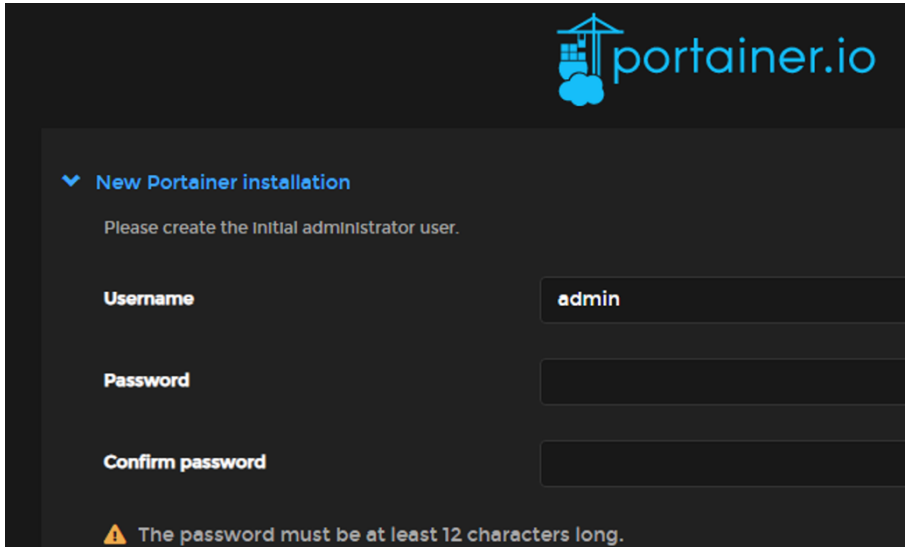
Docker Desktop - deploy Portainer (CLI)

- You can stop/delete the container of the last part if you like
- Now we are going to deploy Portainer as Container Management Tool (this is later used in EI Cloud for deployments)
  - https://docs.portainer.io/start/install/server/docker/wsl
  - First we create a volume to store portainers database
    - *docker volume create portainer_data*

```
C:\>docker volume create portainer_data
portainer_data
```

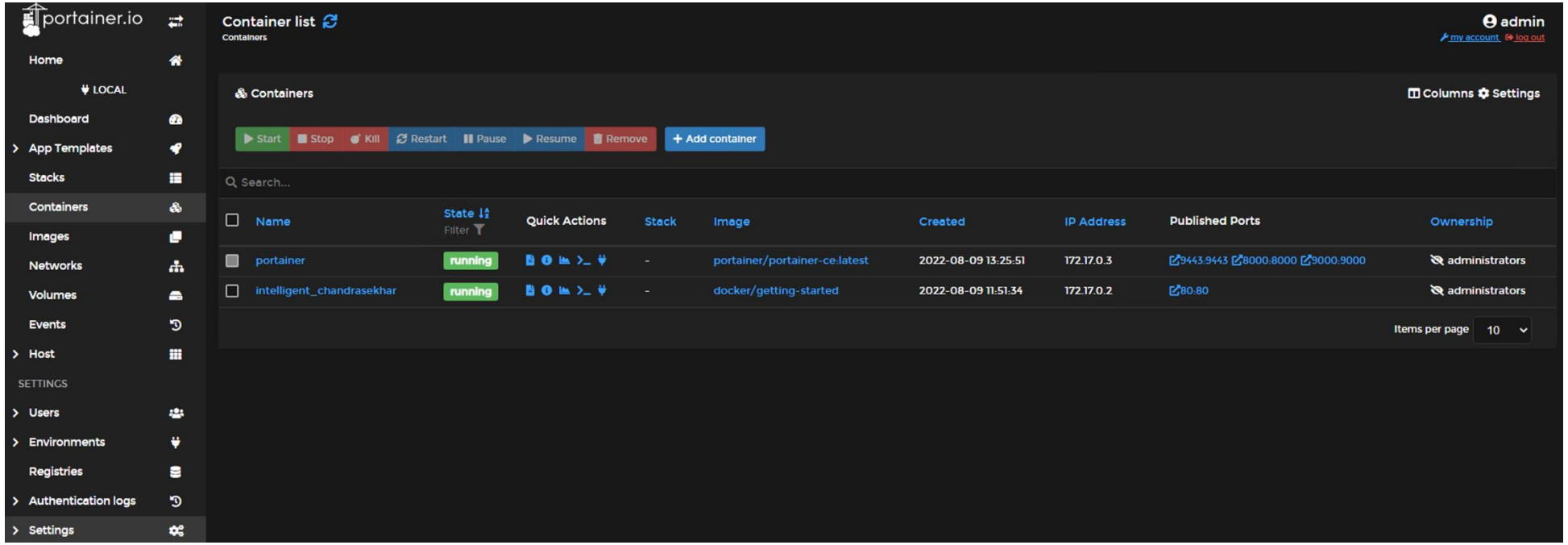
  - Then download and deploy portainer server (community edition)
    - *docker run -d -p 8000:8000 -p 9443:9443 -p 9000:9000 --name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce:latest*
  - open portainer in browser with port 9000 and set up user / password

Port 8000: edge agent functionality
Port 9443: WebUI TLS Port (own certificate will be created while deployment)
Port 9000: WebUI non TLS Port



Docker Desktop - deploy Portainer (CLI)

- After login into portainer you can find your deployed containers in Home → local (environment) → containers



- Make yourself familiar with portainer environment, have a look at containers, networks, volumes, published ports, internal IPs etc...
- We will dive deeper into portainer and deployments in the next labs
- For now it is only a means to the end

Docker Desktop - deploy Node-Red (CLI)

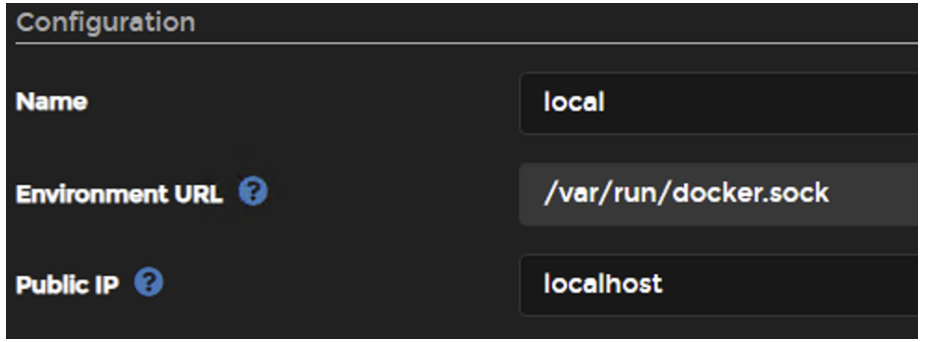
- We are going to deploy Node-Red as webbased Application in CMD
  - *docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered nodered/node-red*
- It is going to be started directly in your cmd window which causes a problem if we close that window
- So we are going to start the container in portainer
  - First terminate node-red in cmd with ctrl+c
  - Navigate in portainer to containers to find node-red (maybe you need to refreh the browser window)
  - “mynodered” should be shown as stopped
  - Select container and start it → starting → refresh → running
  - In Portainer GUI click on port 1880 in published port list
    - Browser opens with 0.0.0.0:1880 and an error message
    - To correct that, navigate to environments → local and put localhost in field Public IP → update environment
    - Open via port again → should be localhost:1880 now

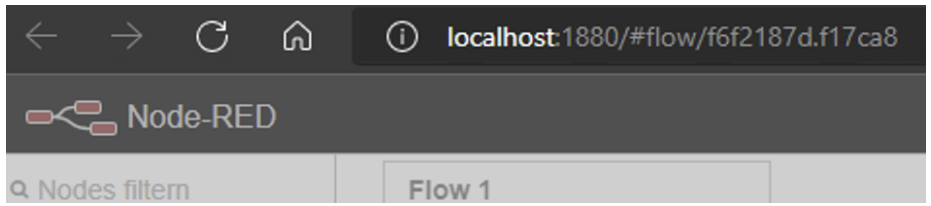
```
---------------------------------------------------------------------
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
---------------------------------------------------------------------

9 Aug 11:42:01 - [info] Server now running at http://127.0.0.1:1880/
9 Aug 11:42:01 - [warn] Encrypted credentials not found
9 Aug 11:42:01 - [info] Starting flows
9 Aug 11:42:01 - [info] Started flows
```

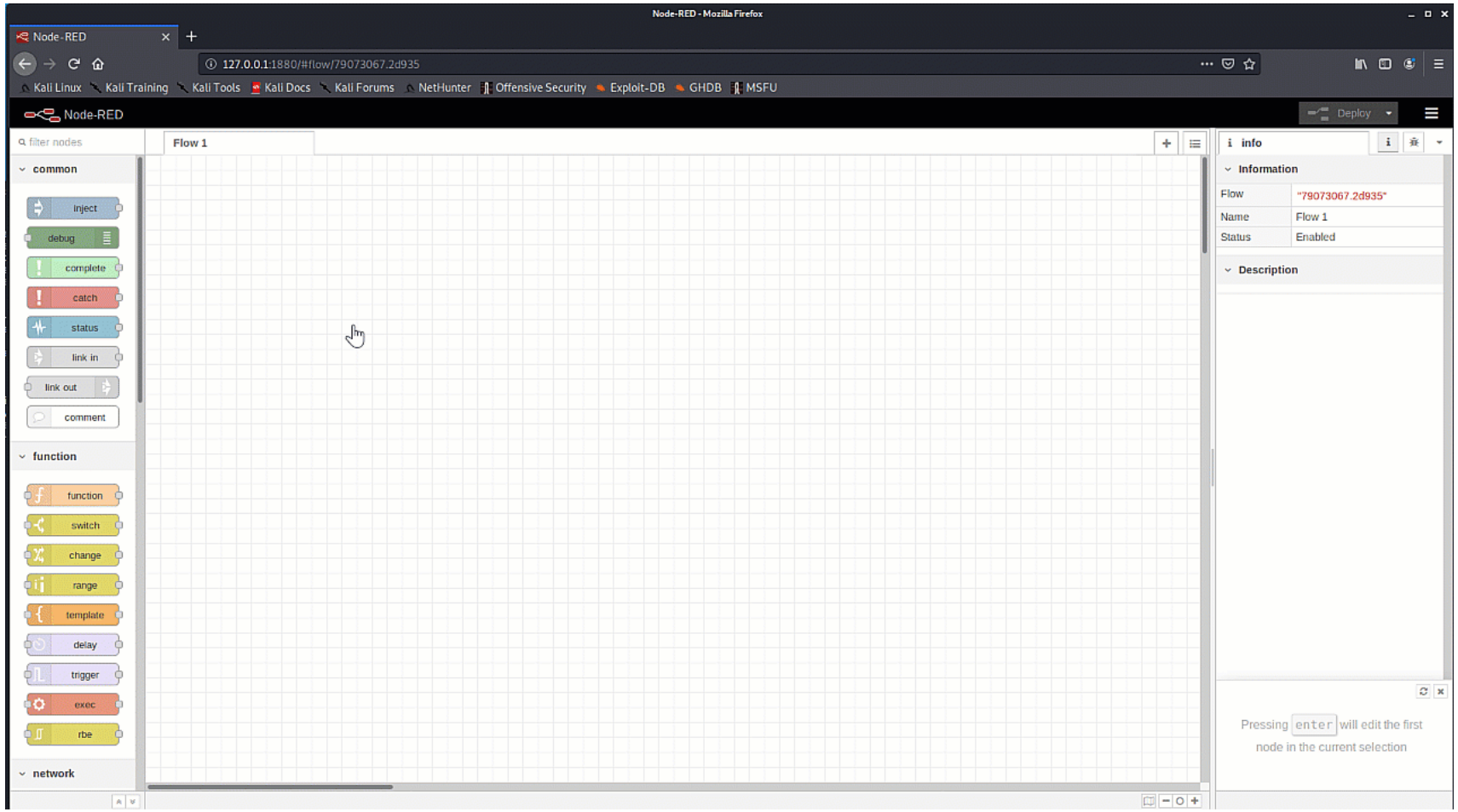




Node-Red - Exercise 1

Node-Red introduction as simple flow

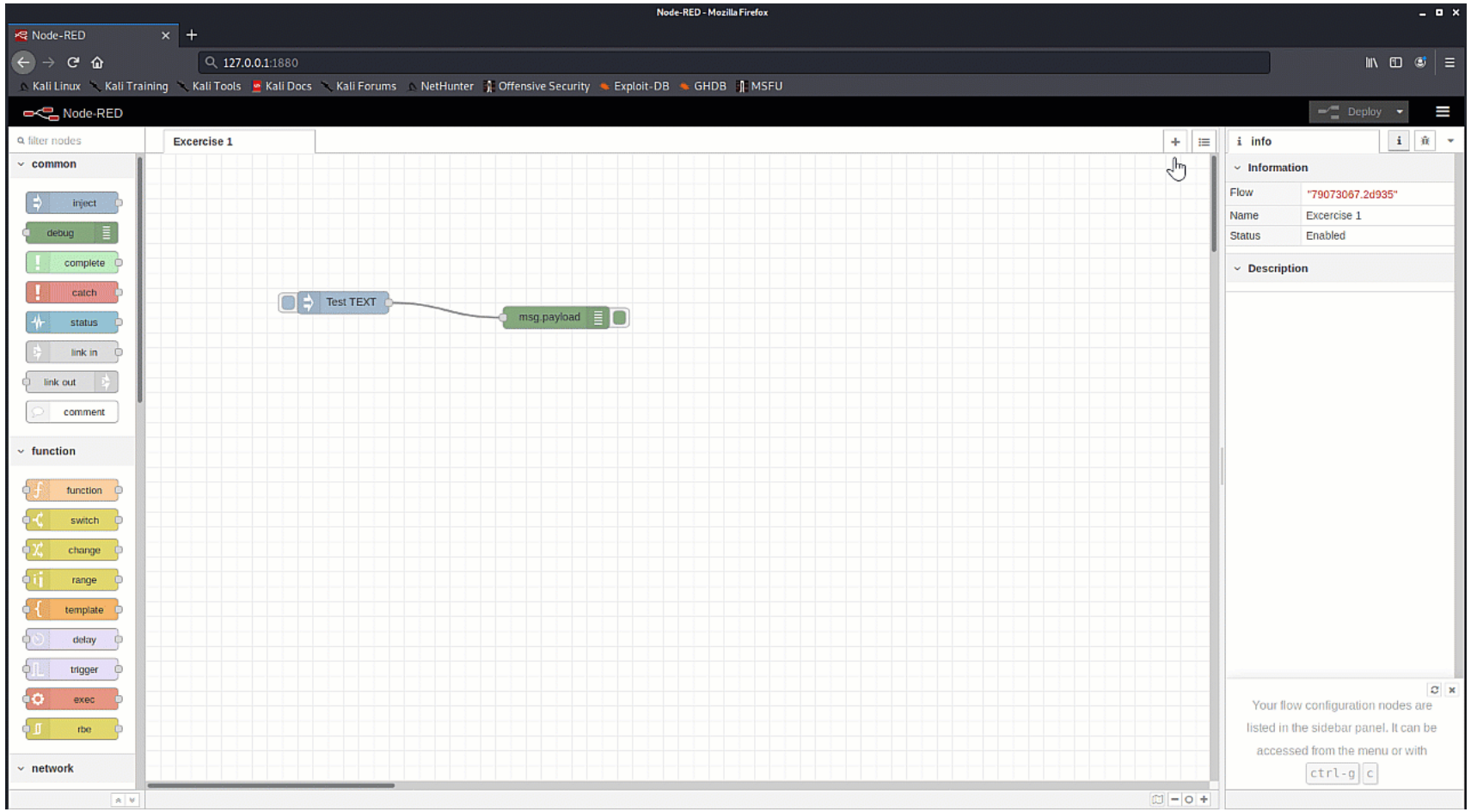
- rename your flow
- drag and drop 2 nodes:
  - **inject**
  - **debug**
- and **connect** them
- change inject node to **„string“ and put in text in payload field**
- switch window to debug
  - → **deploy!**
  - click inject **button**
- text should be in right debug window



Node-Red - Exercise 2

Simple chat with MQTT Broker

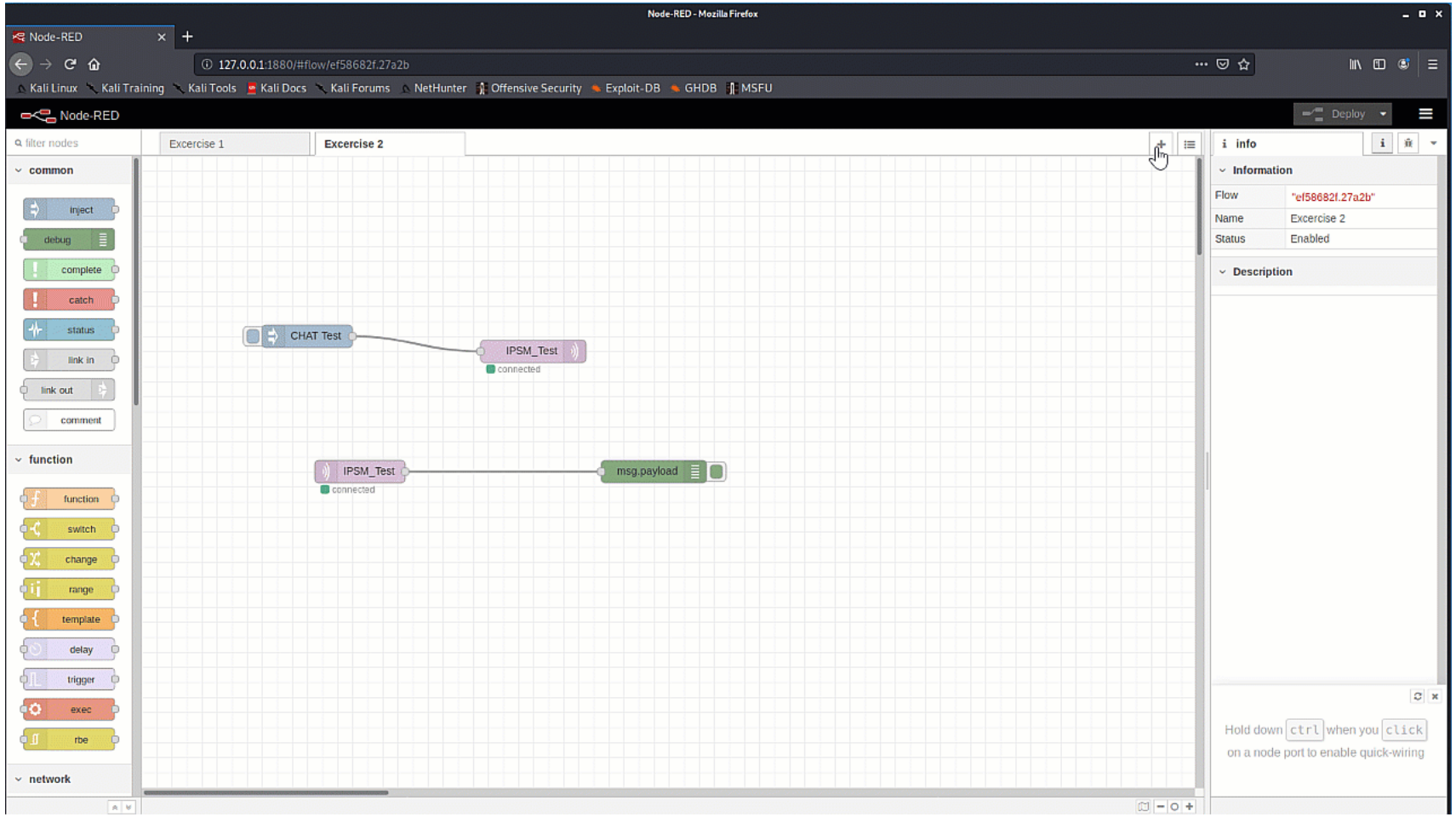
- **Create 2nd flow** and **rename** it
- send a string (inject node) to a public **MQTT** Broker (**publish**) with an unique topic: ***THM/IoTLab/yourname***
- read it (**subscribe**) and show it as text in **debug** window
- example MQTT Broker:
  - ***test.mosquitto.org***
  - → simple chat through MQTT Broker
- Check MQTT Messages in MQTT Explorer



Node-Red - Exercise 3

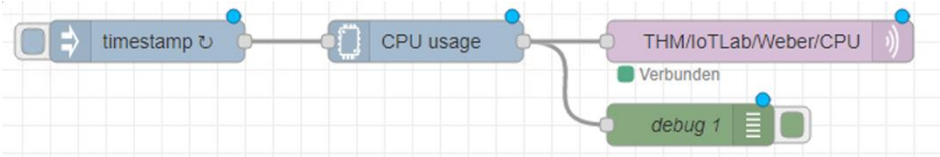
Read CPU overall load

- **Create 3rd flow** and **rename** it
- Install a new node:
  - **node-red-contrib-cpu**
- use new node and display CPU load in debug window with following **settings**:
  - **inject-node with interval of 1 second**
  - **CPU Node: overall usage**
  - **debug node: complete msg object**



Node-Red - Exercise 4

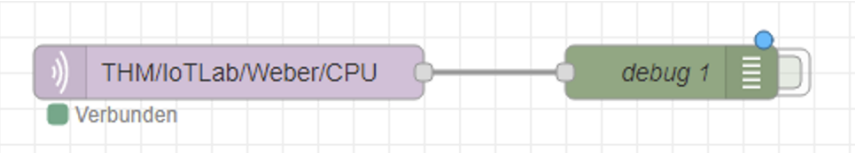
- We are going to send the CPU Data through MQTT back to our flow
  - Use the following MQTT topic: THM/IoTLab/<yourname>/CPU




```
9.8.2022, 14:48:15 node: debug 1
overall : msg : Object
▸{ payload: 0.16666666666666666,
topic: "overall", _msgid:
"db348461e53e00a1" }
```

```
9.8.2022, 14:48:16 node: debug 1
overall : msg : Object
▸{ payload: 0.08333333333333333,
topic: "overall", _msgid:
"bb0265729bd38923" }
```

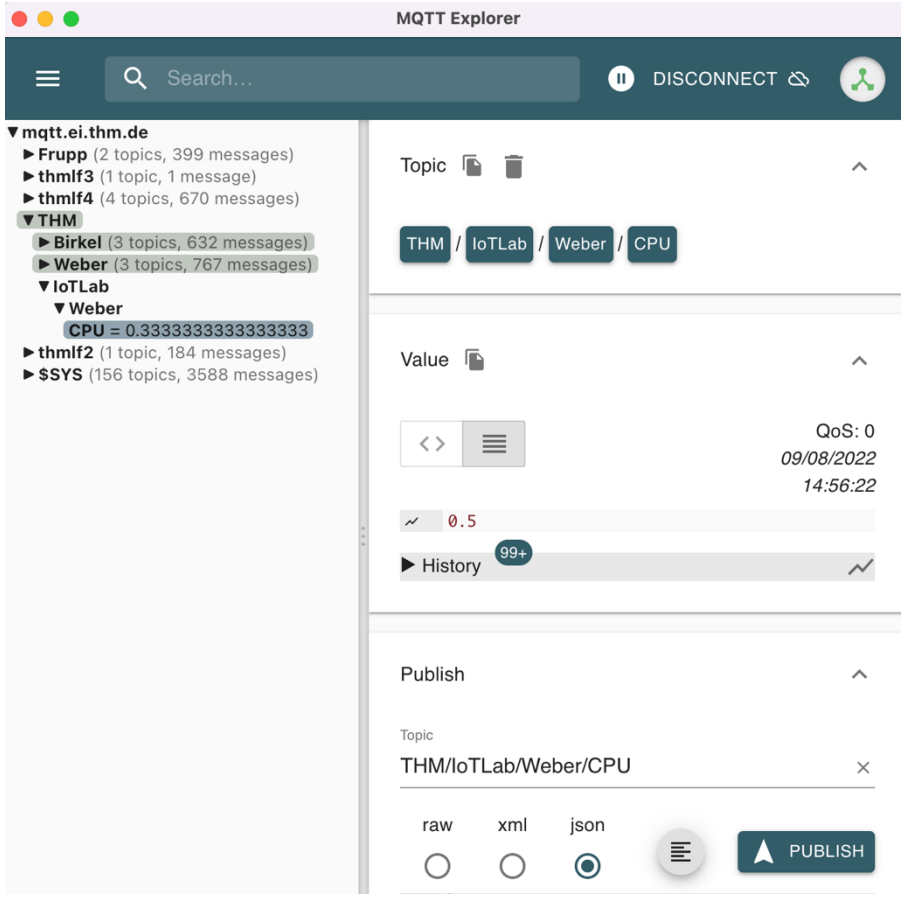
  - Subscribe again and show values in debug
  - Have a look at the complete msg object and compare to the object before sending with MQTT → can you find a difference?




```
9.8.2022, 14:51:29 node: debug 1
THM/IoTLab/Weber/CPU : msg : Object
▸{ topic: "THM/IoTLab/Weber/CPU",
payload: 0.4166666666666667, qos: 2,
retain: false, _msgid:
"6f0e15ada6f5de5a" }
```

```
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"
```

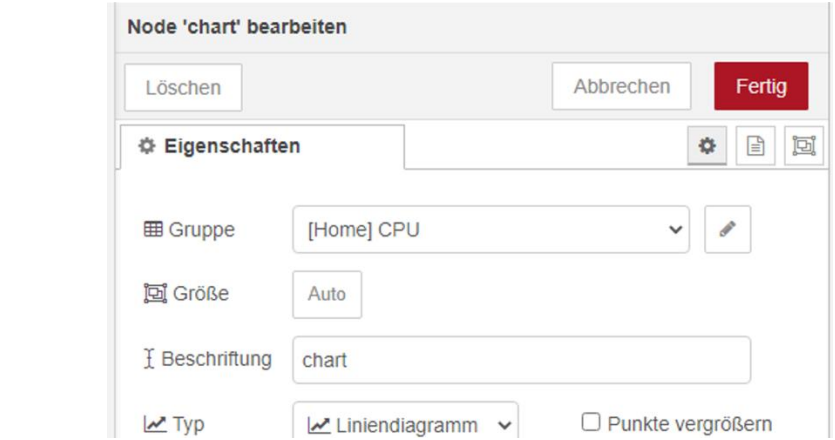
  - Analyse the MQTT Traffic as well with the MQTT Explorer

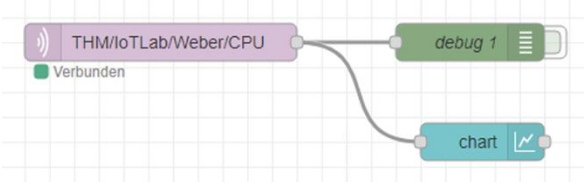


Node-Red - Exercise 5

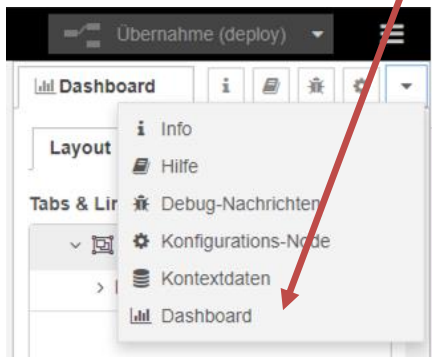
Create Dashboard and show values

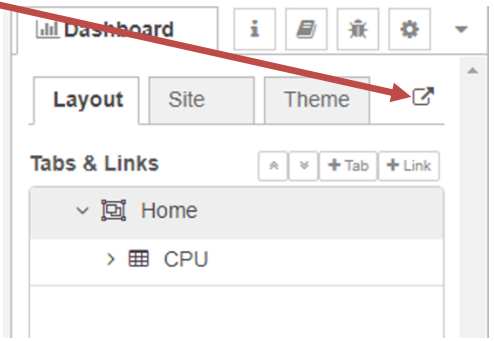
- **Install** dashboard nodes:
  - **node-red-dashboard**
- Use chart node
- Create **new dashboard group (name it CPU) and new ui tab (can be left Home)**
- Connect the subscribing MQTT Node to your chart node

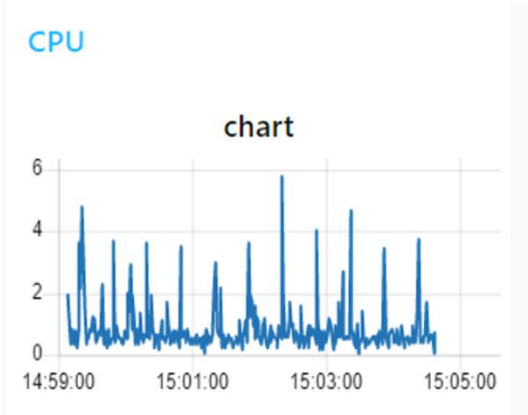




- Open **dashboard** → CPU values should be shown as simple chart







For advanced users

- If you like to deploy portainer as a swarm
  - https://docs.portainer.io/v/ce-2.9/start/install/server/swarm/wsl
  - **docker swarm init**
  - change to writable working directory
  - curl -L https://downloads.portainer.io/portainer-agent-stack.yml -o portainer-agent-stack.yml
  - docker stack deploy -c portainer-agent-stack.yml portainer

  - Login in https://localhost:9443 or http://localhost:9000