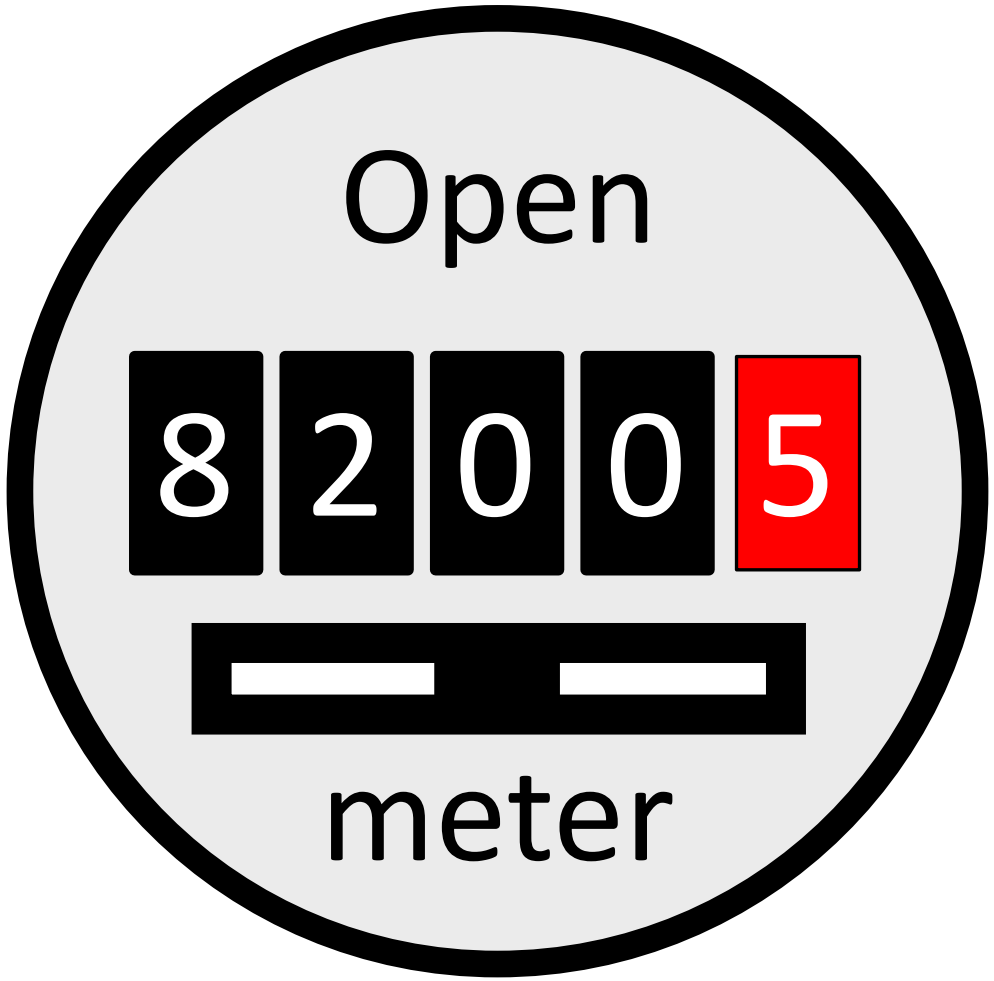

Open
82005
meter

- Opnemen workshop

De 5 geboden

- 100% vrijblijvend & niet werk
- Docent = een digitale enthousiast, not “docker certified professional”
- Be open and share
- Learning by doiing
- Learning by laughing

Agenda:

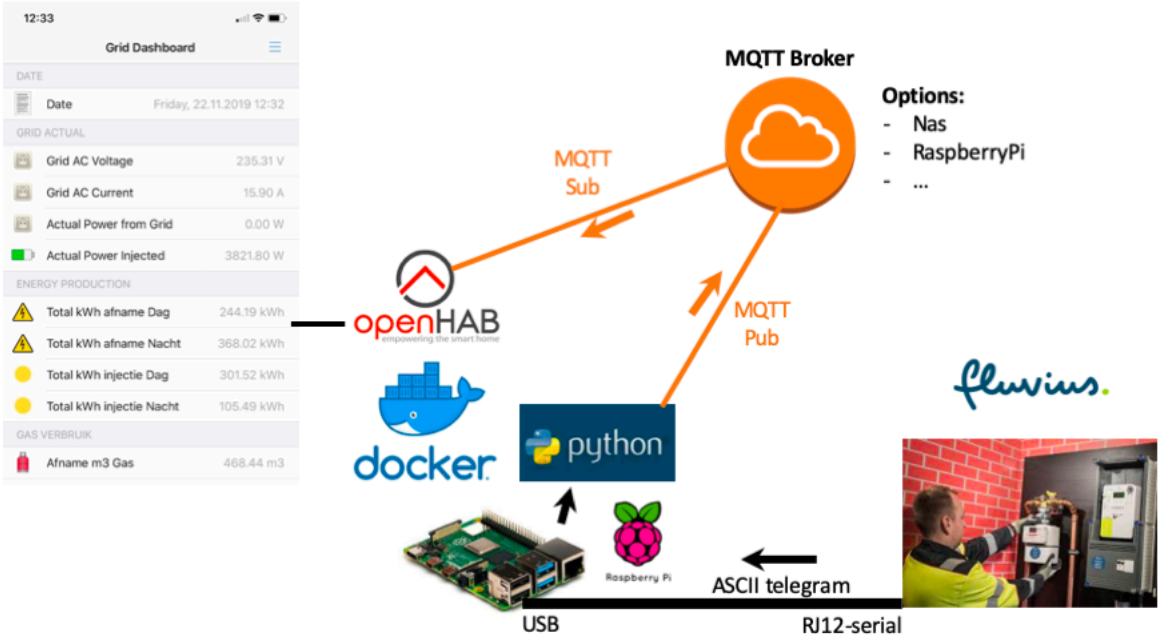
- Workshop 1:
  - What are we going to build ?
  - Intro to docker
- Workshop 2:
  - Docker-compose
  - Lab: Build a smart meter with 1 command

- Agenda Workshop 1:
- Intro:
  - What are we going to build ?
    - Various options: ESP32 or RaspberryPi ? (pro/con + future)
  - The world around us
    - Where and what to learn ?
    - The digital “eco-system”
- Intro to docker
  - Server evolution
  - Docker:
    - Docker theory
    - Basic docker
      - Images versus containers
      - commands: run, ps, ls, detach, it, name, ports, start, stop, rm, arguments
      - Debug: inspect, log
      - Volumes
      - networks
    - Dockerfile
      - Docker build
      - Dockerhub - https://hub.docker.com/

What are we **NOT** going to build ?

History = MVP ☺

https://github.com/tribp/DSMR-Fluvius-MQTT-Openhab



Sub-optimal for:

- Architecture
- Reproduction
- Flexibility
- Abscraction & isolation
- Future evolution

Side info DSMR:

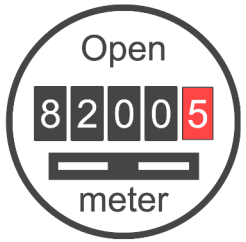
- Belgium = not Holland
- Interface = pré-internet technology
- RS232 = inverted
- Extra HW needed:
  - uController, RPI, PCB,…
  - Or RS232-USBB cable

‘containers are ephemeral’

Ephemerality is the concept of things being transitory, existing only briefly. Typically the term ephemeral is used to describe objects found in nature, although it can describe a wide range of things, ...
Wikipedia

What are we going to build ?

Final Project



WWW

Docker-compose.yaml

Grafana

grafana

influxdb

influxdb

openHAB

openhab

MQTT Broker

mosquitto

mqtt_broker

MQTT Client

python

dsmr_fluvius

fluvius.

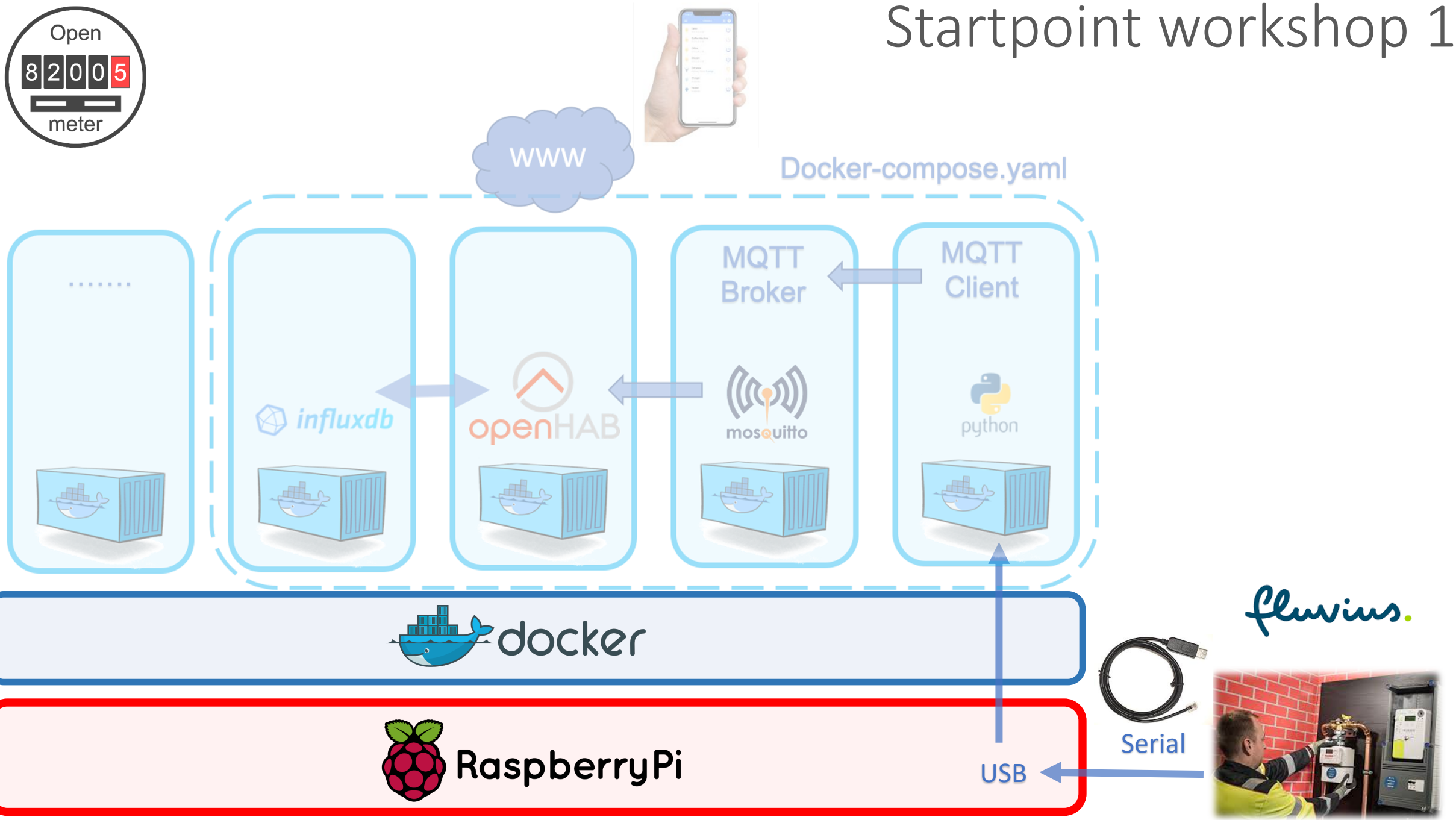
docker

RaspberryPi

Serial

USB

Startpoint workshop 1



What are we going to build ?

Various options

ESP32

RPI

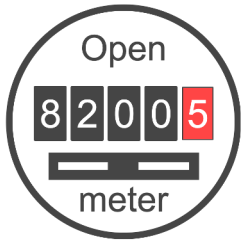
Pro:
- 6 euro
- Dongle -isch
- Flexible
- Low-energy – 0.1 Watt

Con:
- Housing
- PCB + RS232 invertion
- (slightly) more complex Architecture

Pro:
- All-in one approach possible
- Standard HW – software platform
- One touch provisioning

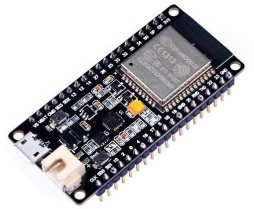
Con:
- Power needed @DSMR
- Power consumption – 4 Watt
- SD card – lifecycle
- 60 euro

Final Project-ESP32



WWW

Docker-compose.yaml



Grafana

influxdb

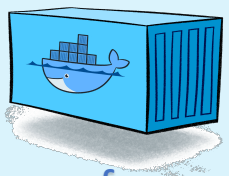
openHAB

MQTT Broker

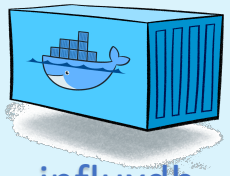
mosquitto

MQTT Client

python

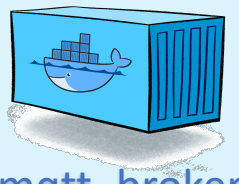


grafana

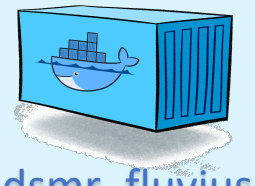


influxdb

openhab



mqtt_broker



dsmr_fluvius

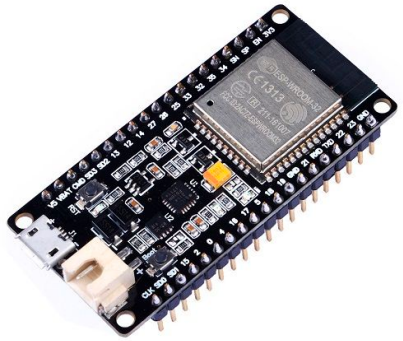
fluvius.

docker

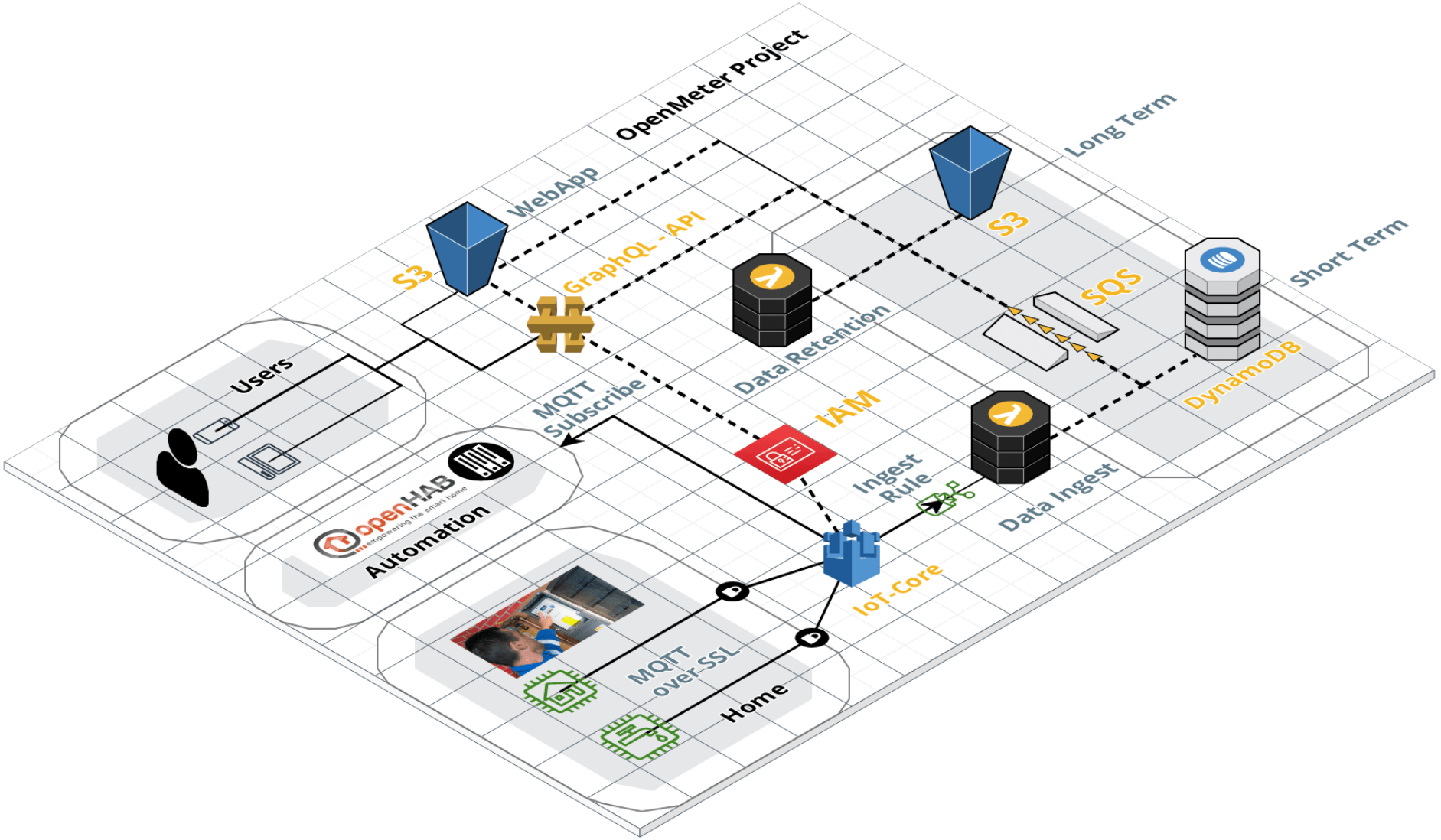
Serial

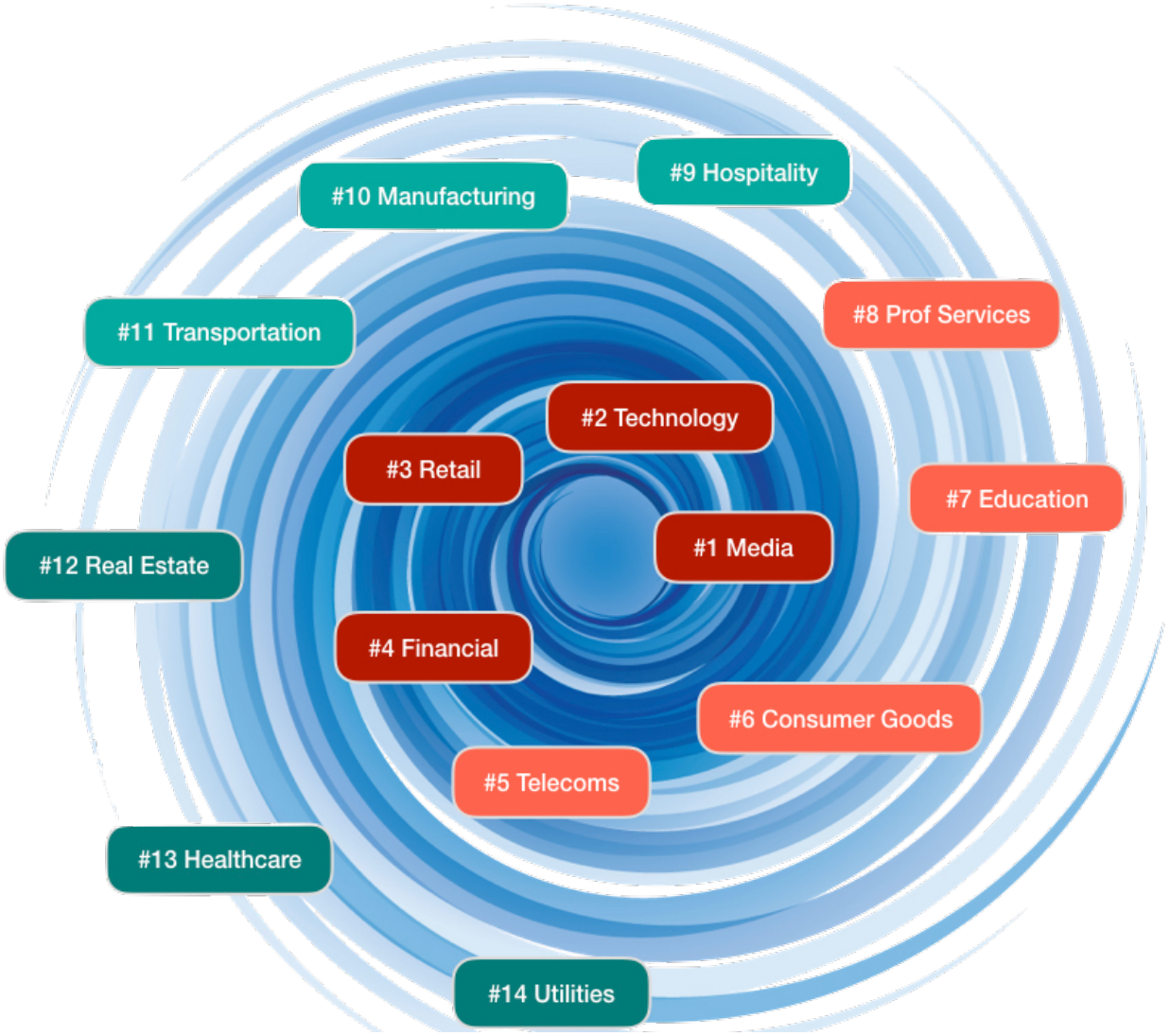
USB

RaspberryPi



Possible architecture with esp32

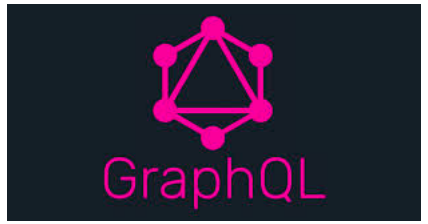


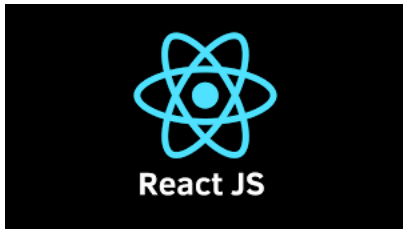


The world around us ?

5 year

What to learn ?





What to learn and where ?

Udemy

Linux Academy

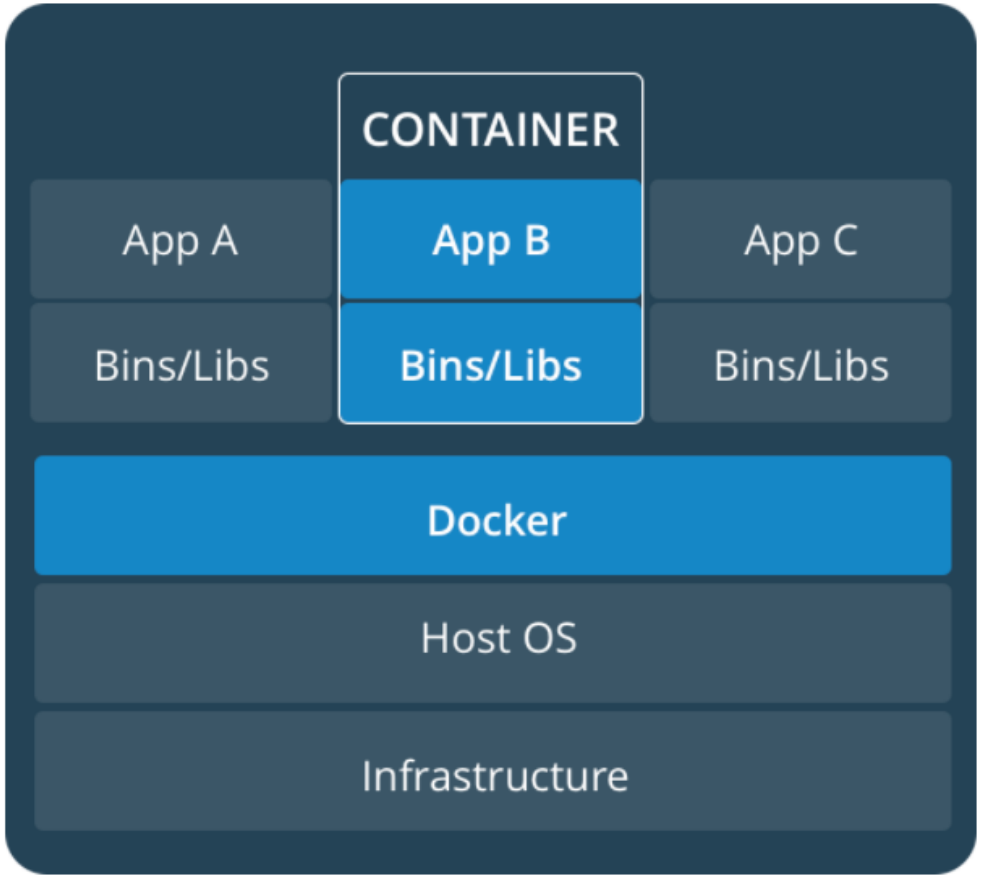
INE

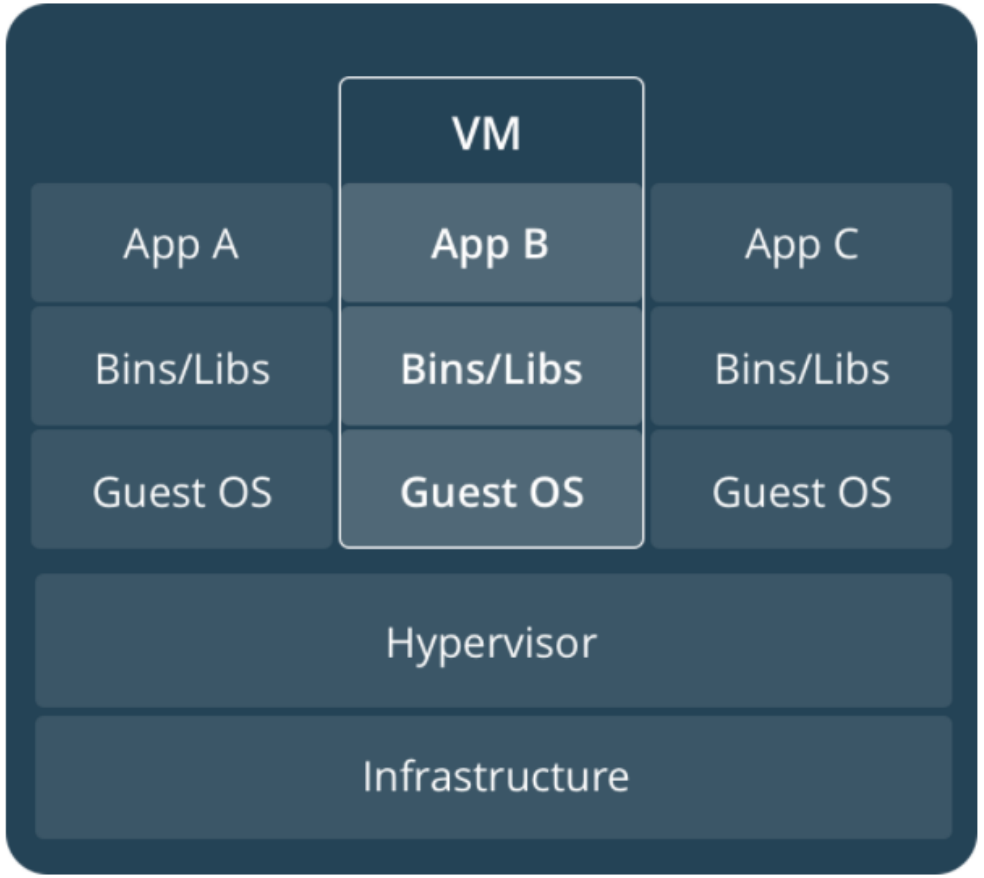
CISCO DEVNET

Intro:

- Intro to docker
  - Server evolution
  - Docker:
    - Docker theory
    - Basic docker
      - Images versus containers
      - commands: run, ps, ls, detach, it, name, ports, start, stop, rm, arguments
      - Debug: inspect, log
      - Volumes
      - networks
    - Dockerfile
      - Docker build
      - Dockerhub - https://hub.docker.com/

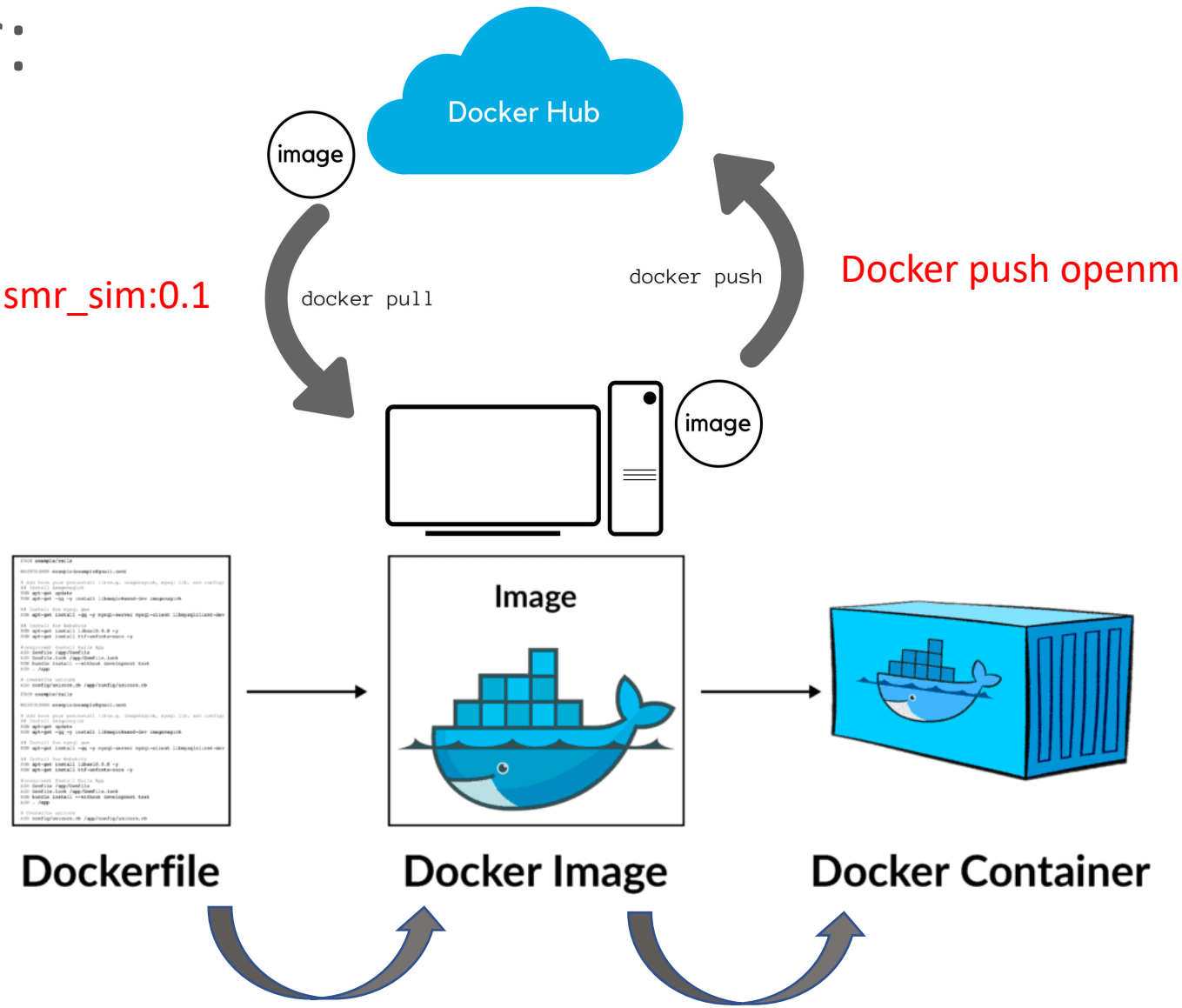
Intro: Server evolution: Docker vs VM





Intro to docker:

openmeter/dsmr_sim



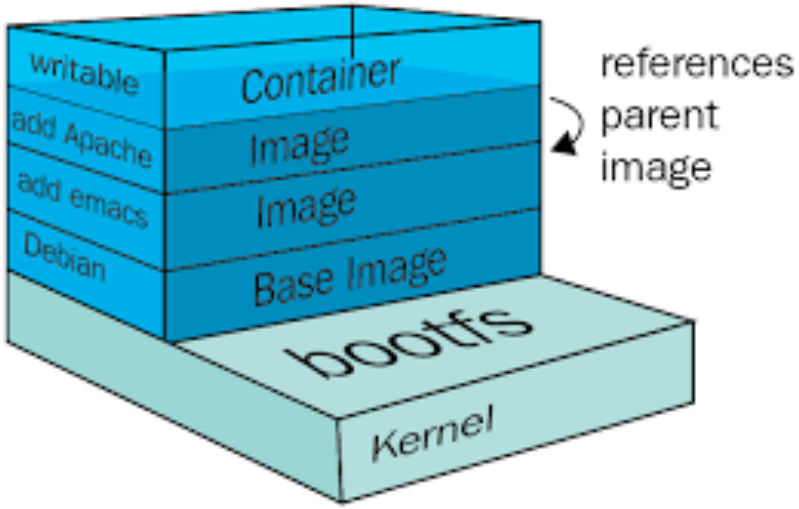
Docker pull openmeter/dsmr_sim:0.1

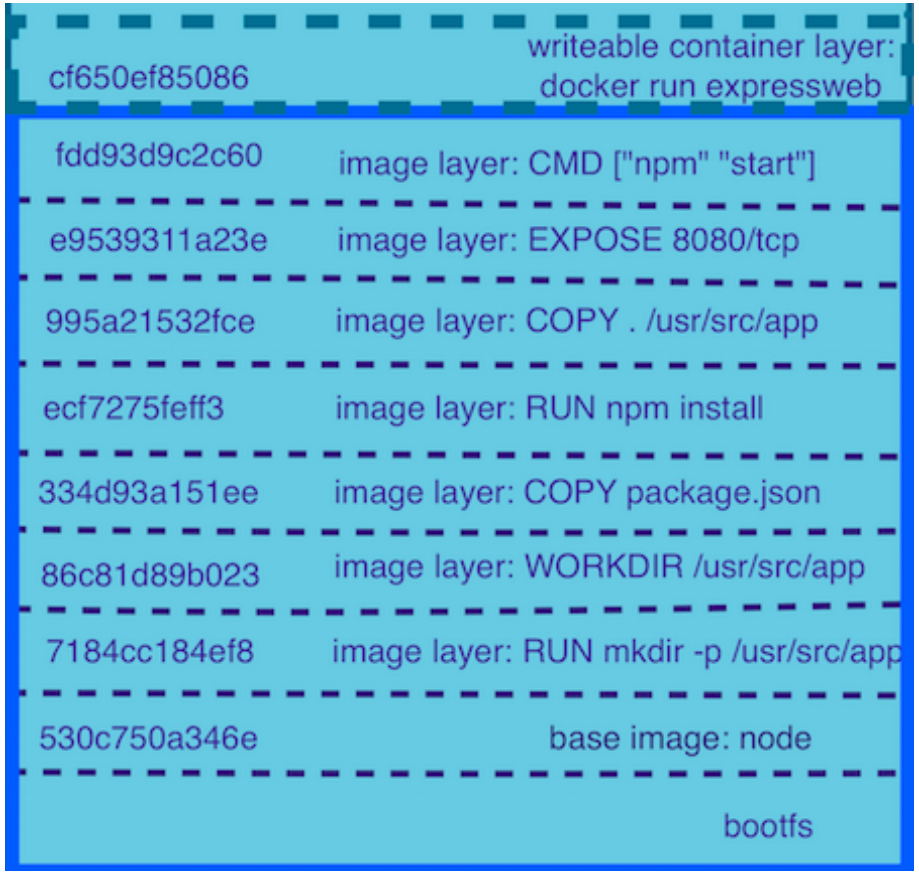
Docker push openmeter/dsmr_sim:0.1

Docker build openmeter/dsmr_sim:0.1 .

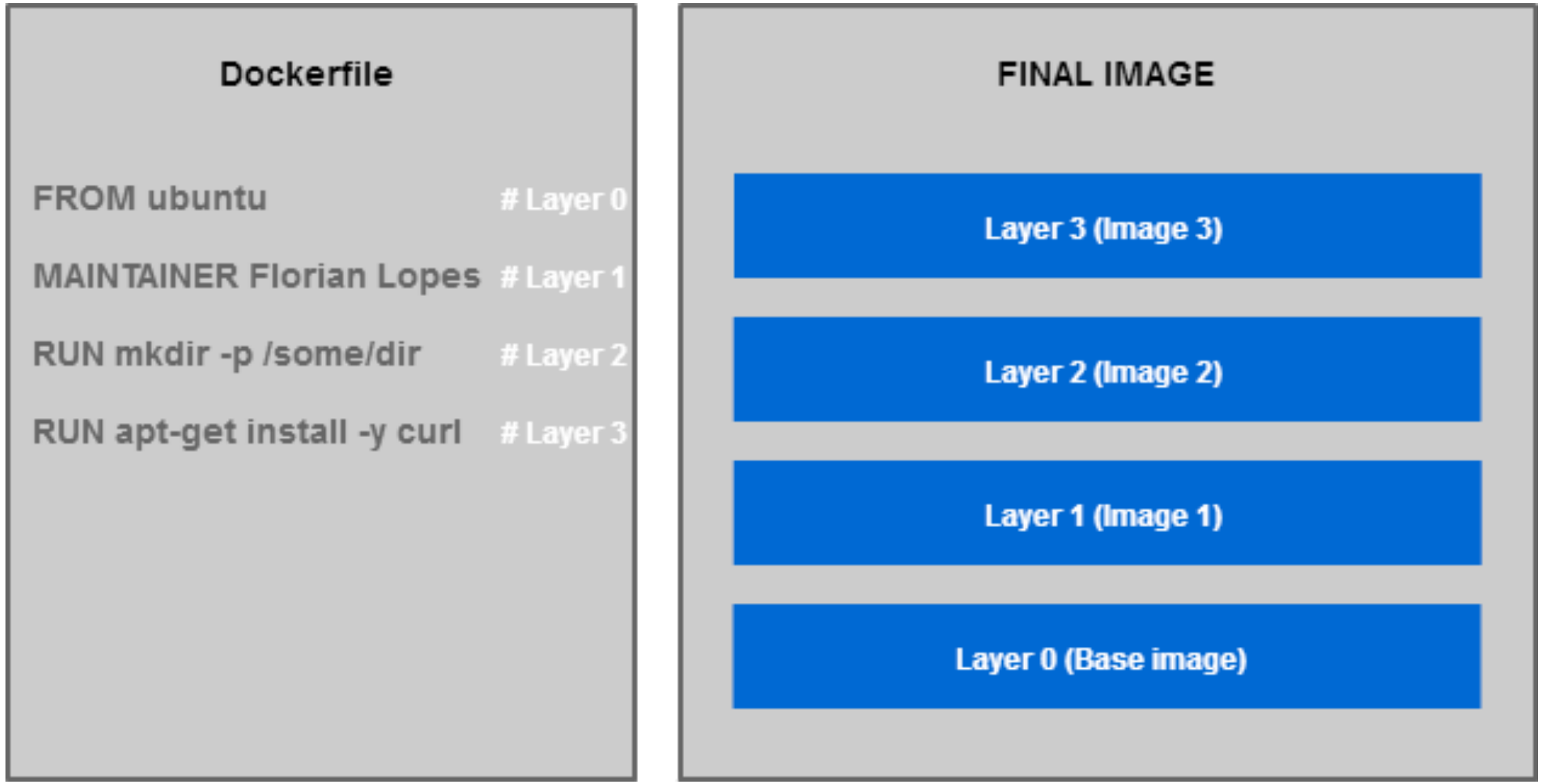
Docker run –d openmeter/dsmr_sim:0.1

Docker image layers:





Docker image layers:



Lab: check docker installation

- “docker –version”
- “docker ps”
- “docker image ls”
- “docker container ls” === “docker ps”
- “docker container ps –a”

Lab: run your first container

- “docker run nginx”
- “docker ps”
- -> probleem
- “docker ps -a”

```
pi@raspberrypi:~ $ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS
                    NAMES
cffd320e597b        nginx               "/docker-entrypoint.…"   33 seconds ago      Exited (0) 14 seconds ago
                    laughing_galois
```




Lab: run your first container

- Look at https://hub.docker.com
  - Official images
  - Platforms
  - Tags
  - Dockerfile
  - cmd

Lab: run your first container

- “docker run –d nginx”
- “docker ps”
- -> probleem
- “docker ps -a”

```
pi@raspberrypi:~ $ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
57eba35ad870        nginx               "/docker-entrypoint.…"   7 seconds ago       Up 5 seconds        80/tcp              fervent_cannon
```

Lab: run your first container

- Optimize our container
  - Give it a name -> -name mySVR
  - Remove it after run -> --rm
  - Define port -> -p 8080:80

- “docker run –d –rm –name mySVR –p 8088:80 nginx”
- “docker ps”
- Check in chrome !

Lab: run your first container

- Start – stop
  - Docker stop mySVR
  - Check in chrome
  - Docker start mySVR
- Deleting
  - Docker stop 57eba35ad870
  - Docker rm 57eba35ad870
  - Docker rmi nginx

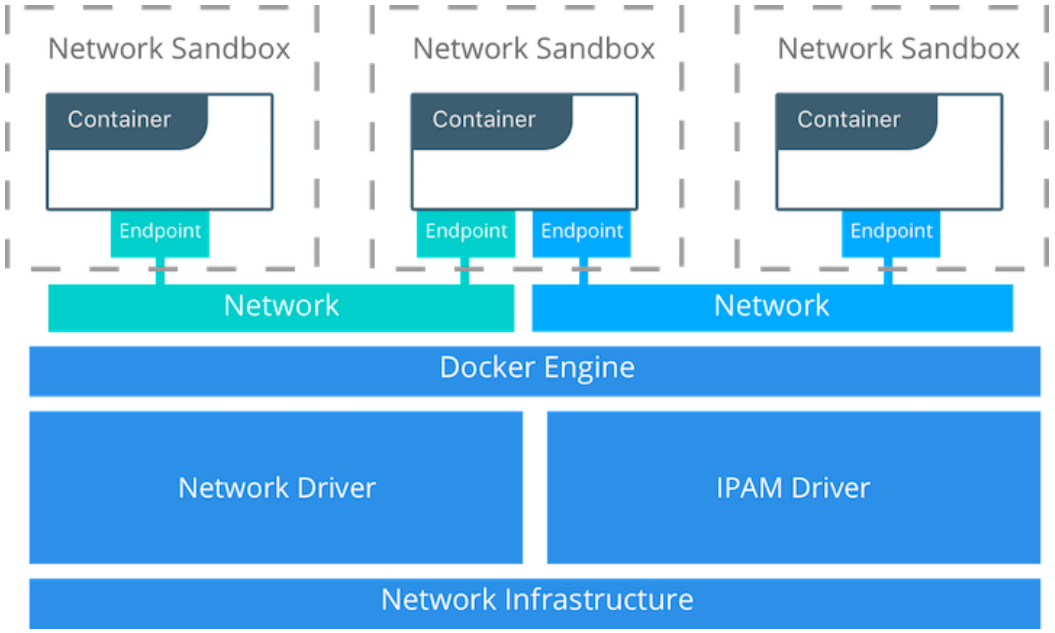
Lab: Make your own webserver

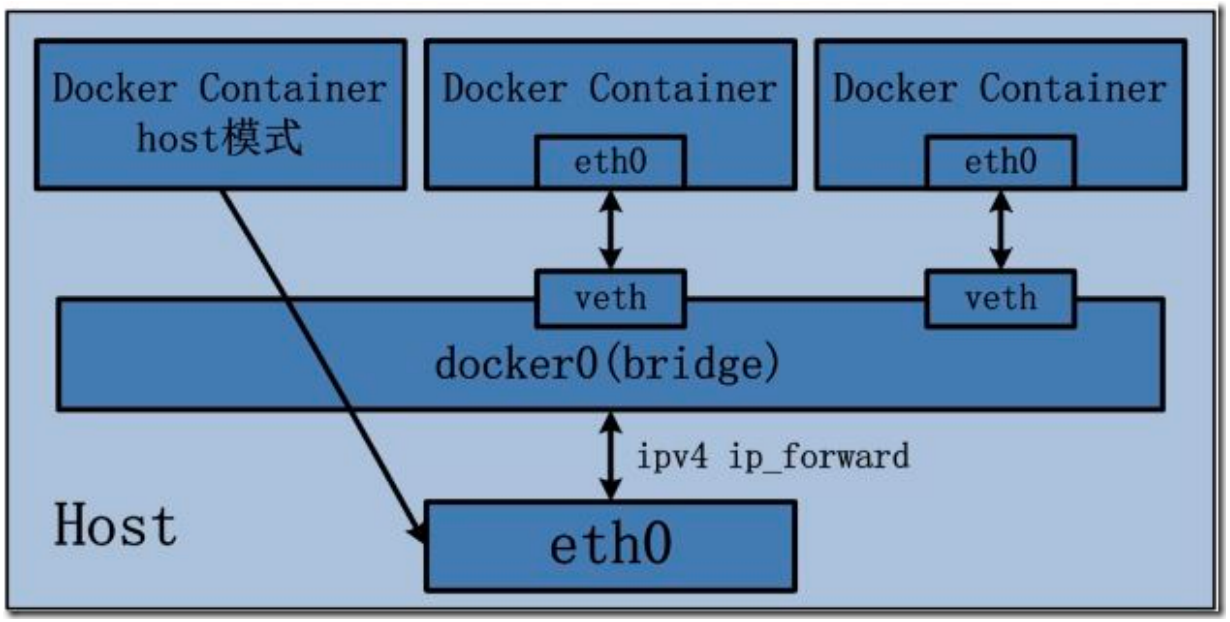
- Start nginx in detach mode
  - Docker run –d –rm –name myWS –p 8088:80 nginx
  - Check in chrome
- How to connect a bash window to container === ssh like
  - Docker exec –it myWS bash
  - Cd /usr/share/nginx/html
  - Nano index.html
  - Apt-get update
  - Apt-get install nano
  - Nano index.html -> change to …..
  - Check in chrome
  - GRRRR if container gone -> changes gone
    - Solution 1 = volumes
    - Solution 2 = dockerfile

Lab: Mapping volumes for your own webserver

- Start nginx in detach mode with volumes
  - Mkdir test
  - Cd test
  - Pwd -> gives /home/pi/test
  - Copy content from index.html
    - https://github.com/openmeter/openmeter-RPI-get-ready/blob/master/src/index.html
  - Nano index.html + paste + save
  - docker run -d --name myWS --rm -p 8088:80 -v /home/pi/test:/usr/share/nginx/html nginx
  - Check in chrome
  - Stop container / change index.html / run container ☺

Docker networks





Lab: docker networks

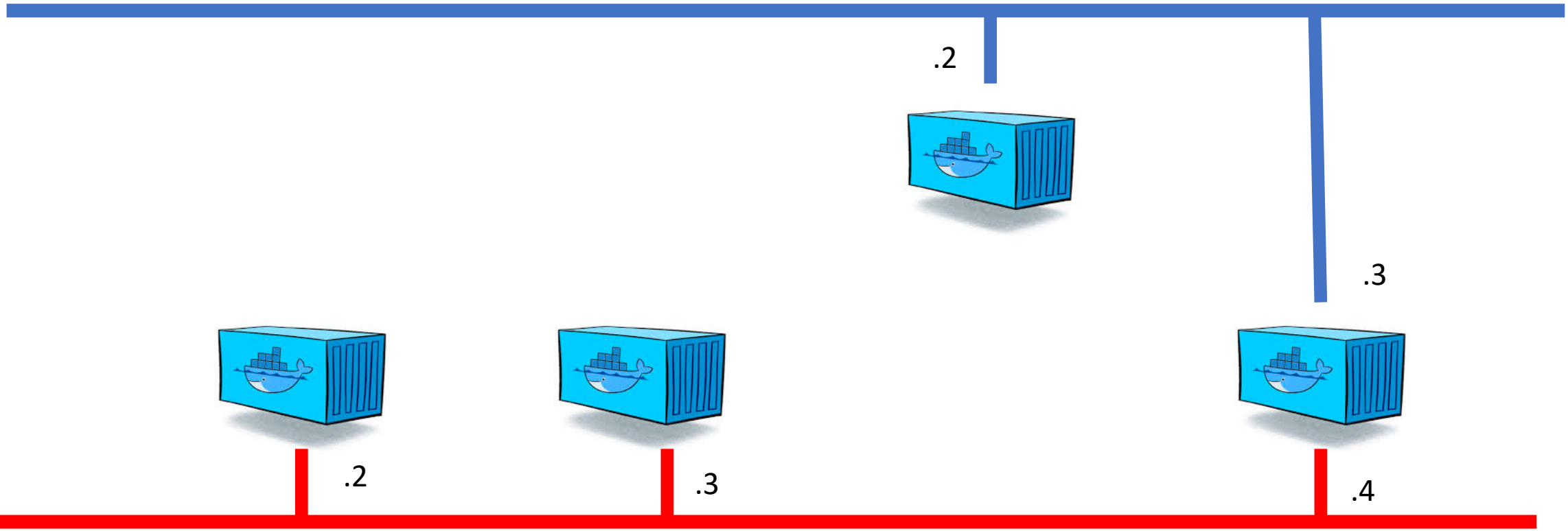
- Types of network
  - null
  - Host
  - bridge
- Listing available networks
  - Docker network ls
- Inspecting a network
  - Docker network inspect bridge
- Creating a network
  - Docker network create backend
  - Docker network ls
  - Docker network inspect backend

Lab: todo (https://docs.docker.com/network/network-tutorial-standalone/)

- Create 2 networks
  - bridge network
  - alpine-net network
- Create 4 containers
  - docker run --dit --name alpine1 –net alpine-net alpine ash
  - docker run --dit --name alpine2 –net alpine-net alpine ash
  - docker run --dit --name alpine3 alpine ash
  - docker run --dit --name alpine4 –net alpine-net alpine ash
    - Docker network connect bridge alpine4
- Inspecting network
  - Docker network inspect bridge // docker network inspect alpine net
- Attach and ping
  - Docker attach alpine1 // ping –c 2 alpine2 // ping –c 2 alpine3 // ping –c 2 alpine4
  - Lookup ip addr alpine3 // ping x.y.z.w // (ctr p ctrl q)
  - Docker attach alpine4 // ping alpine3 // ping x.y.y.z

172.17.0.0/16

Network: bridge



Network: alpine-net

x.y.z.u

Remark:

- Later with docker compose
  - Network: alpine-net
    - -> result = containerName_alpine-net

Lab: We bouwen 2 eerste containers van finale oplossing. Maar voorlopig ‘manueel”

- To do:

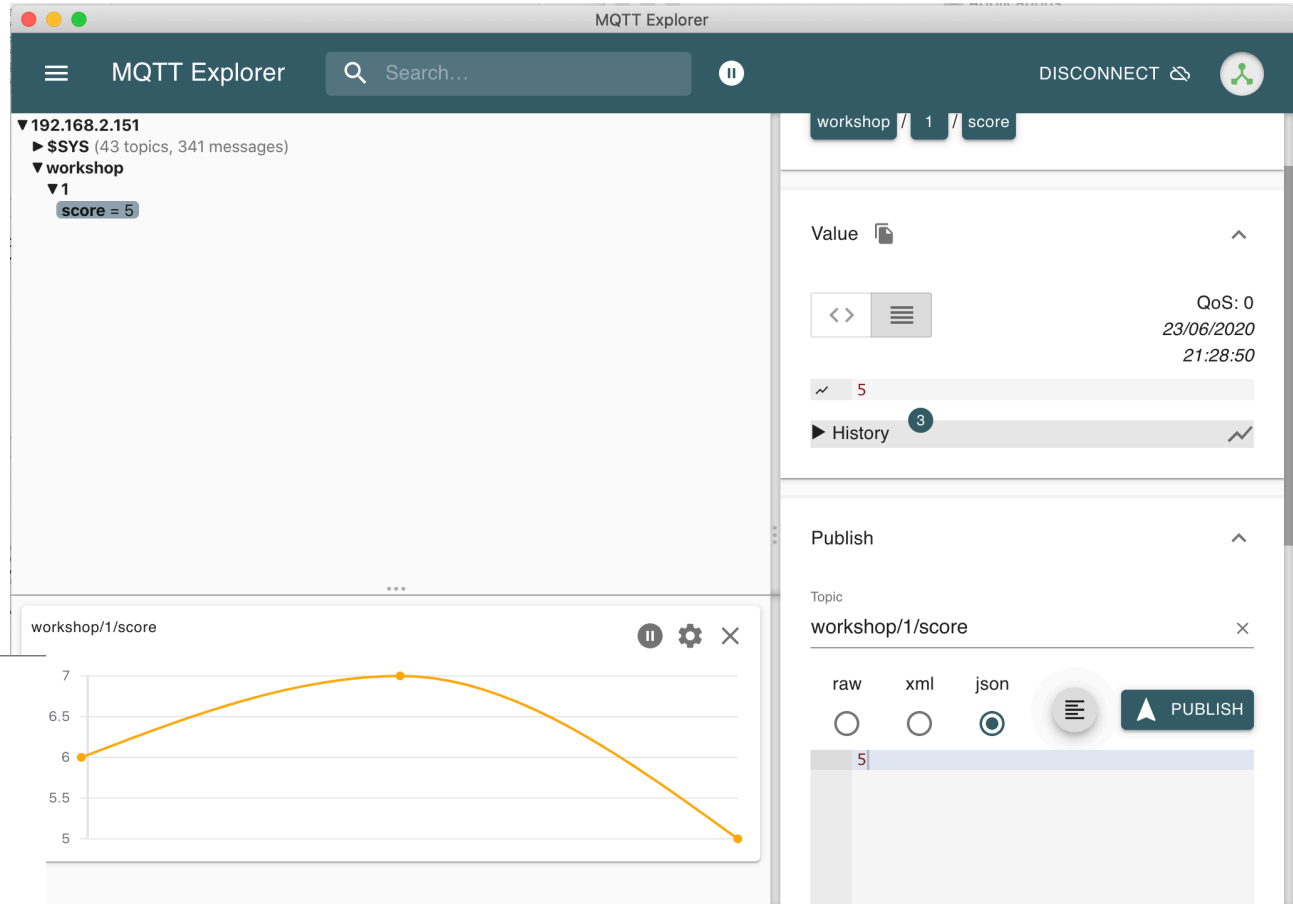
1. MQTT broker
2. MQTT Client (DSMR simulator)

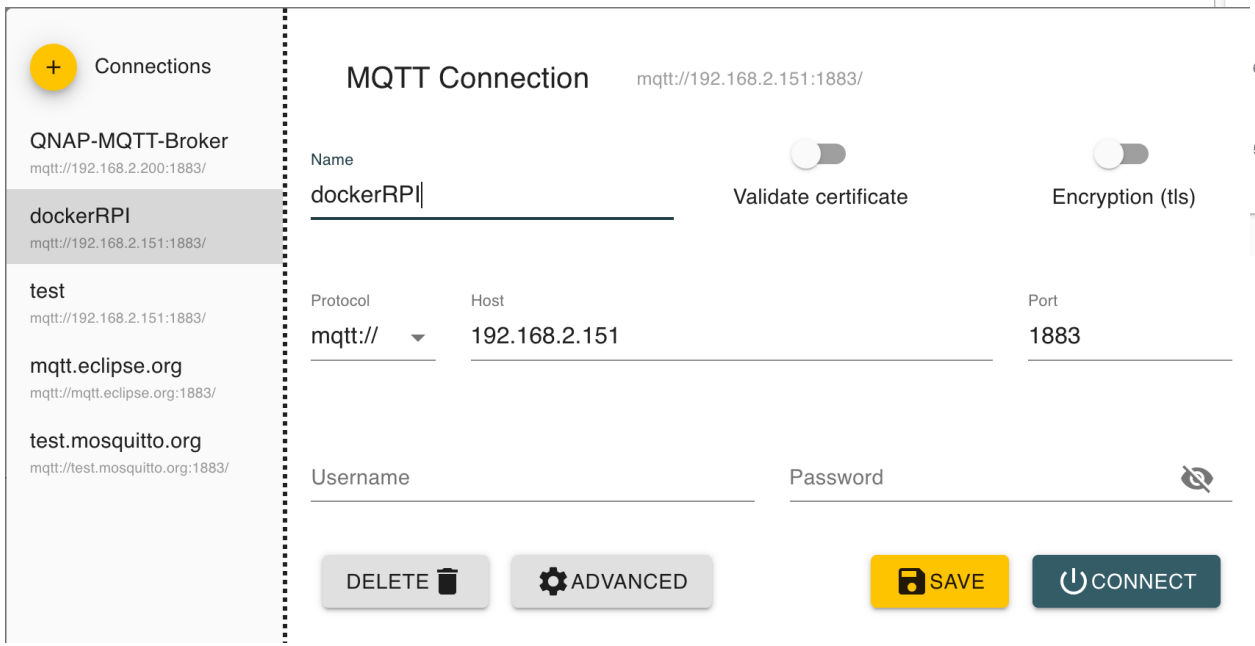
Lab: MQTT Broker”

- Zie docker hub -> eclipse-mosquitto
- docker run -d -p 1883:1883 --name mqtt_broker -v /openmeter/data/mqtt_broker/config:/mosquitto/config eclipse-mosquito
- Stop container + delete container -> we doen dit opnieuw maar met “extra” voor ‘network’
- docker run -d -p 1883:1883 --name mqtt_broker --net backend -v /openmeter/data/mqtt_broker/config:/mosquitto/config eclipse-mosquito
- Test:
  - Docker network ls
  - Docker network inspect backend -> noteer ip adres van mqtt_broker
  - docker run --rm --net backend alpine ping -c3 192.168.48.4
  - docker run --rm --net backend alpine ping -c3 mqtt_broker
    - Docker image ls -> see size alpine and broker !!!

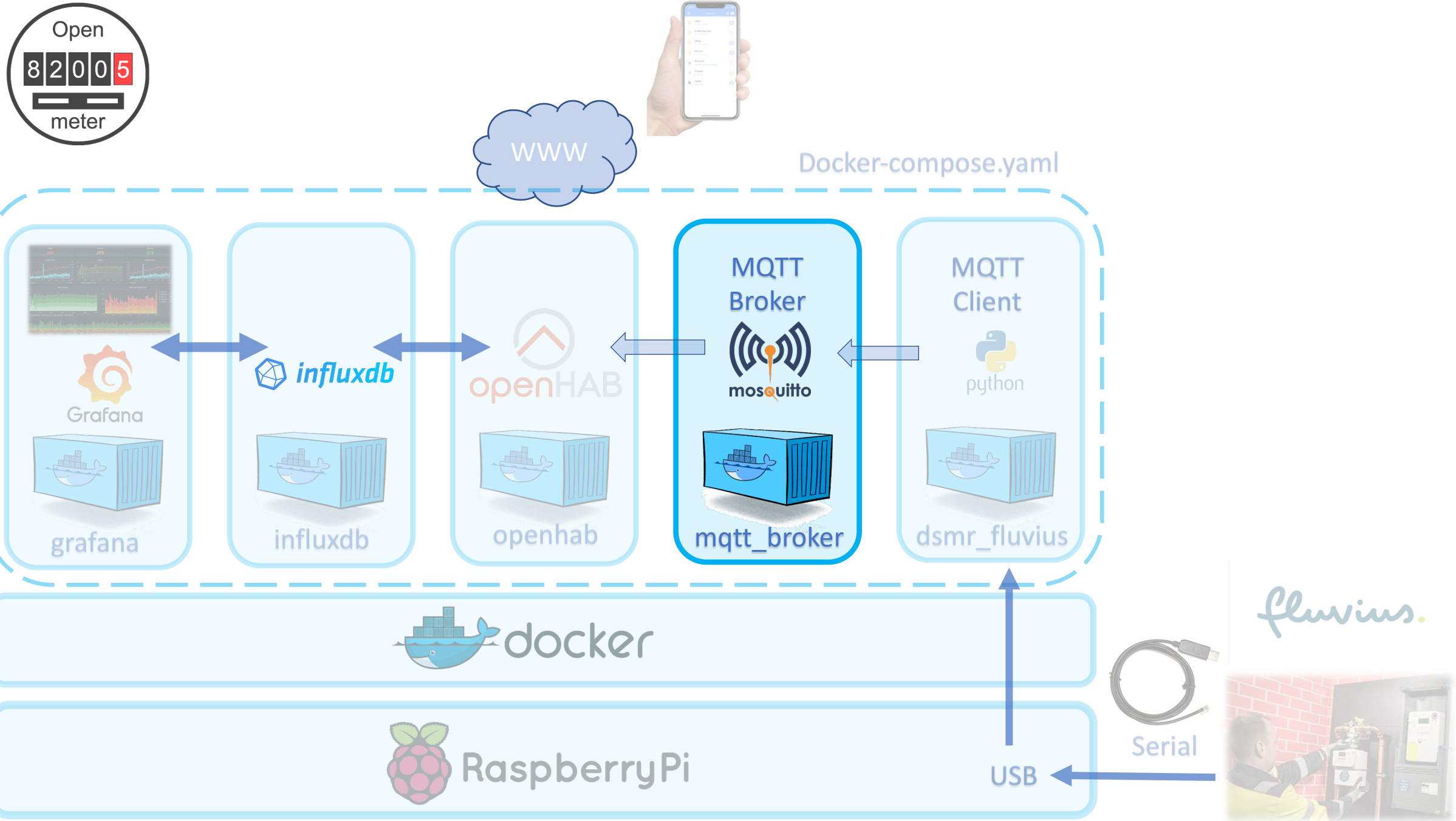
Lab: MQTT Broker”

- Test with “MQTT explorer






CHAMPAIN


Open
82005
meter
www
Docker-compose.yaml
Grafana
grafana
influxdb
influxdb
openHAB
openhab
MQTT
Broker
mosquitto
mqtt_broker
MQTT
Client
python
dsmr_fluvius
docker
RaspberryPi
Serial
USB
fluvius.

Lab: MQTT-Client (DSMR simulator)

- First we will make container that simulates digital meter

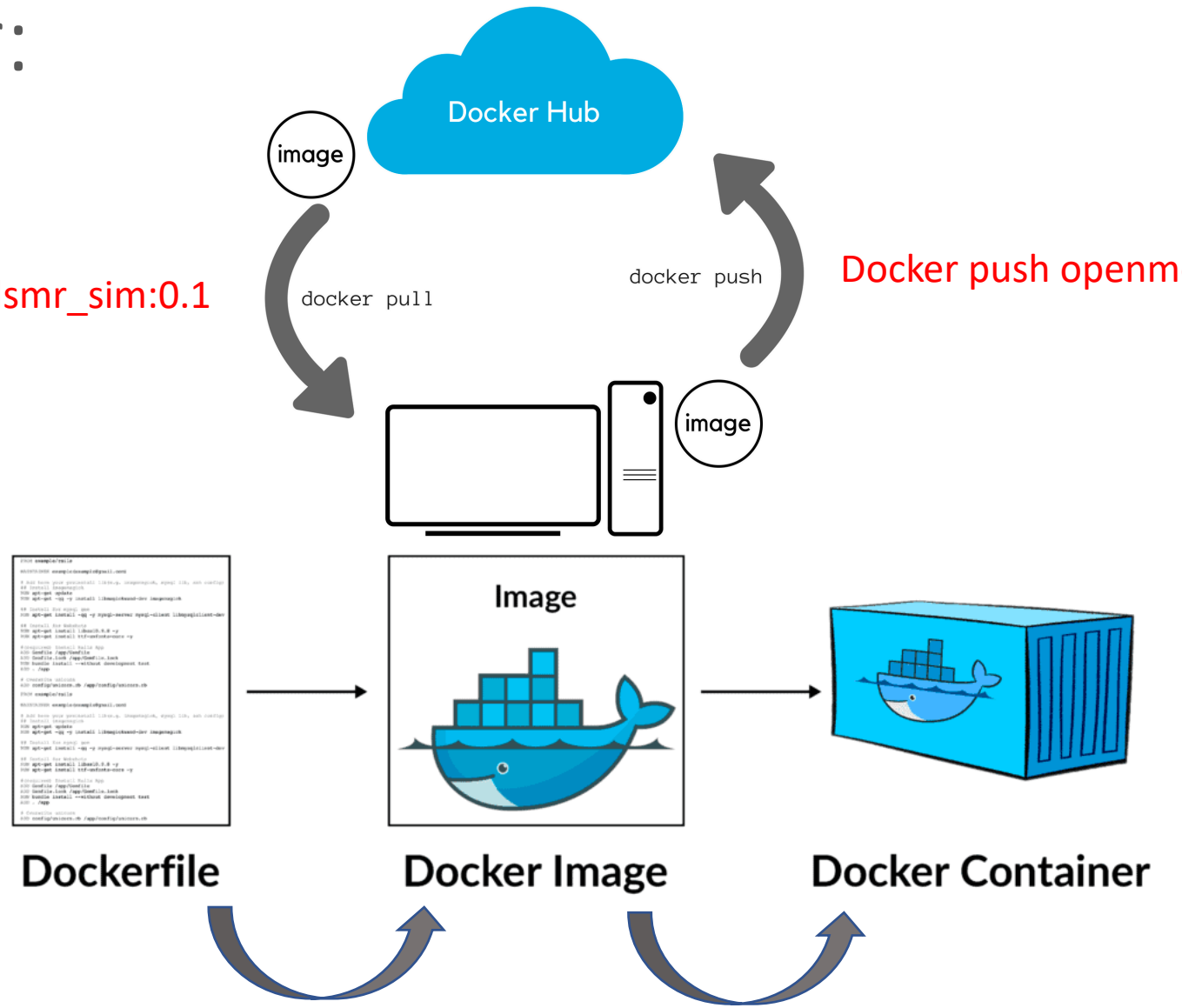
1. We will create and pimp the container manually
2. We will test is (MQTT explorer)
3. We destroy is
4. We automate the building with a dockerfile
5. We will push the dockerfile to “docker Hub’
6. We will recreate a container that will pull it from docker HUB

Lab: MQTT-Client (DSMR simulator)

- The manual proces:
- docker run -dit --name dsmr_sim -v /openmeter/data/dsmr_sim:/dsmr_sim ubuntu:18.04 bash
- docker exec -it dsmr_sim bash
- Pimp it:
  - Check if /dsmr_sim exists
  - apt-get update
  - apt-get install -y python3.7
  - apt-get install -y python3-serial python3-pip python3-yaml
  - pip3 install paho-mqtt
  - Apt-get nano
  - Cd /dsmr_sim
  - Nano p1_fluvius_smartmeter_sim.py
  - Copy content + save + close
  - Python3 p1_fluvius_smartmeter_sim.py

Intro to docker:

openmeter/dsmr_sim



Docker pull openmeter/dsmr_sim:0.1

Docker push openmeter/dsmr_sim:0.1

Docker build openmeter/dsmr_sim:0.1 .

Docker run –d openmeter/dsmr_sim:0.1

Lab: MQTT-Client (DSMR simulator)

- The automated proces:
- Dockerfile
- docker build –t /openmeter/dsmr_sim:0.3 .
- Docker push /openmeter/dsmr_sim:0.3
- docker run -d --name dsmr_sim --net backend openmeter/dsmr_sim:0.3
- Test it

Enquete invullen