sudo docker run busybox echo “hello from busybox”

Docker release notes:

<https://docs.docker.com/engine/release-notes/>

Docker tutorial for beginner:

<https://docker-curriculum.com/>

Docker Overview:

<https://docs.docker.com/get-started/overview/>

Docker command line reference:

<https://docs.docker.com/engine/reference/commandline/ps/>

Eclipse-mosquito – Official Image

<https://hub.docker.com/_/eclipse-mosquitto>

mosquito.conf man page

[https://mosquitto.org/man/mosquitto-conf-5.html#](https://mosquitto.org/man/mosquitto-conf-5.html)

InfluxDB Official Image

<https://hub.docker.com/_/influxdb>

<https://www.youtube.com/watch?v=0CpHwszFjUY>

Manage InfluxDB security

<https://docs.influxdata.com/influxdb/v1.8/administration/security/>

docker ps - list all containers that are running

docker ps -a - list all containers that we ran before

docker rm $(docker ps -a -q -f status=exited)  
docker ps -af "name=intelligent\_agnesi"

docker images - list all images within docker

docker run busybox --name oi\_handover echo "hello from busybox"  
docker rm oi\_handover  
docker run -it --name OI\_handover busybox sh

sudo docker run -itd --name=MQTTbroker\_oi --restart=always -p 1886:1883 -p 9006:9001 -v /home/marcino/dockerimg/mosquitto/mqtt/config/mosquitto.conf:/mosquitto/config/mosquitto.conf -v /home/marcino/dockerimg/mosquitto/mqtt/data -v /home/marcino/dockerimg/mosquitto/mqtt/log eclipse-mosquitto:1.6

docker logs -f MQTTbroker\_oi  
docker ps -af "name=intelligent\_agnesi"  
sudo docker rm -f MQTTbroker\_oi

**Installing docker on Ubuntu 21.10 Impish:**

sudo apt-get update

sudo apt upgrade

sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common

wget https://download.docker.com/linux/ubuntu/gpg

sudo apt-key add gpg

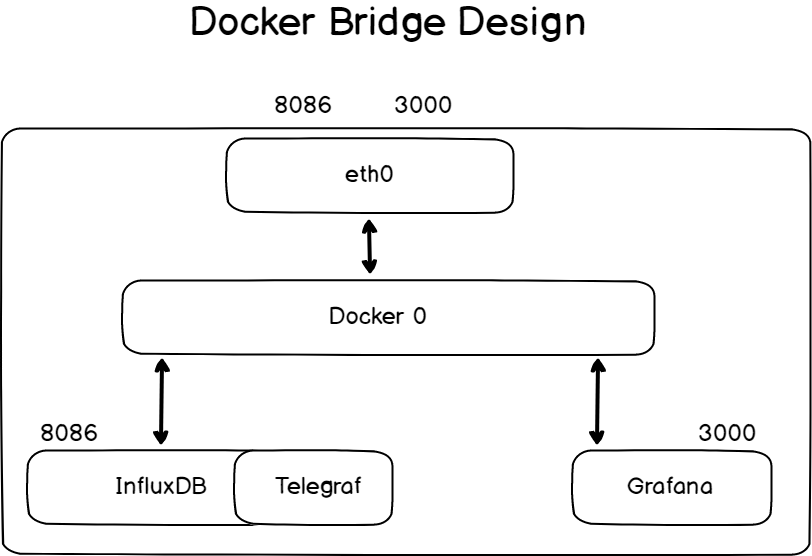
sudo echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee -a /etc/apt/sources.list.d/docker.list

sudo apt update

sudo apt-get install docker-ce docker-ce-cli containerd.io

sudo systemctl start docker

docker -v

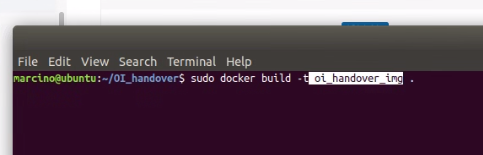


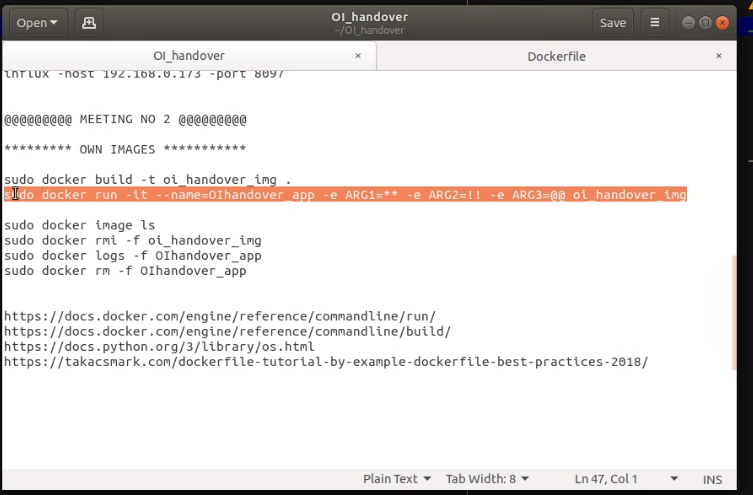
Docker build:

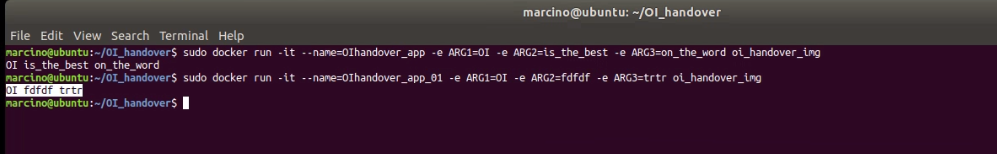




Alpine Linux – smaller amount of memory needed as for python.









\*\*\*\*\*\*\*\*\* OWN IMAGES \*\*\*\*\*\*\*\*\*\*\*

sudo docker build -t oi\_handover\_img . - t is tagname, location of Dockerfile (oi\_handover\_img MUST be in same folder as Dockerfile)

docker build -t image-name -f Dockerfile2 . If Dockerfile isn't named Dockerfile, you can use the -f flag to give the name of the Dockerfile to build.

sudo docker run -it --name=OIhandover\_app -e ARG1=\*\* -e ARG2=!! -e ARG3=@@ oi\_handover\_img

sudo docker run --rm -it ubuntu /bin/bash

sudo docker run -itd --restart=always --name BT1\_DewPointSensor -e MQTTUSERNAME=mqttbroker -e MQTTPASSWORD=LoopEdge.123 -e MQTTHOST=172.17.0.2 -e MQTTPORT=1883 -e MQTTSUBSCRIBETOPIC=PPS\_L1/Assembler\_No1/BT1 -e INFLUXUSERNAME=TomF -e INFLUXPASSWORD=Internet1! -e INFLUXHOST=172.17.0.5 -e INFLUXPORT=8086 -e INFLUXPROCESSDB=PLPPS001\_PROCESS\_DB\_ASSEMBLER dev\_point\_sensor\_img

sudo docker run -itd --restart=always --name RotTable\_VibAccelSensor -e MQTTUSERNAME=mqttbroker \

-e MQTTPASSWORD=LoopEdge.123 -e MQTTHOST=172.17.0.2 -e MQTTPORT=1883 \

-e MQTTSUBSCRIBETOPIC=’PPS\_L1/Assembler\_No1/RotaryTable/Accel/#’ \

-e INFLUXUSERNAME=TomF -e INFLUXPASSWORD=Internet1! -e INFLUXHOST=172.17.0.5 \

-e INFLUXPORT=8086 -e INFLUXPROCESSDB=PLPPS001\_PROCESS\_DB\_ASSEMBLER \

-e INFLUXBATCHSIZE=1000 -e INFLUXFLUSHINTERVAL=100 -e INFLUXJITTERINTERVAL=20 \

vib\_accel\_sensor\_img

sudo docker run -itd --restart=always --name RotTable\_VibAccelSensor -e MQTTUSERNAME=mqttbroker -e MQTTPASSWORD=LoopEdge.123 -e MQTTHOST=172.17.0.2 -e MQTTPORT=1883 -e MQTTSUBSCRIBETOPIC="PPS\_L1/Assembler\_No1/RotaryTable/Accel/#" -e INFLUXUSERNAME=TomF -e INFLUXPASSWORD=Internet1! -e INFLUXHOST=172.17.0.5 -e INFLUXPORT=8086 -e INFLUXPROCESSDB="PLPPS001\_PROCESS\_DB\_ASSEMBLER" -e INFLUXBATCHSIZE=1000 -e INFLUXFLUSHINTERVAL=100 -e INFLUXJITTERINTERVAL=20 vib\_accel\_sensor\_img

sudo docker run -itd --restart=always --net PPS\_Network --ip 172.10.0.6 --name ValveAnomaly valve\_anomaly\_img

for InfluxDB ver.2.x for INFLUXPROCESSDB the declared bucket name must be given (e.g. ‘bucket1’).

sudo docker image ls

sudo docker images - list of images

sudo docker rmi -f oi\_handover\_img - delete docker image

sudo docker logs -f *container\_id* - shows logs from

sudo docker rm -f OIhandover\_app - delete docker container

sudo docker ps - list running docker processes

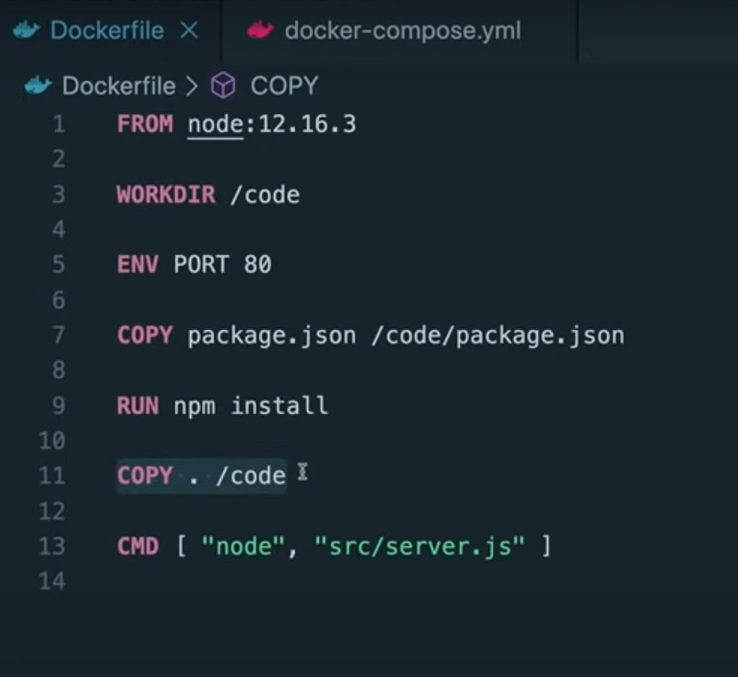
https://docs.docker.com/engine/reference/commandline/run/

https://docs.docker.com/engine/reference/commandline/build/

https://docs.python.org/3/library/os.html

<https://takacsmark.com/dockerfile-tutorial-by-example-dockerfile-best-practices-2018/>

Example of docker file:



MY TESTS:

Well explanation how to set up mosquitto container on docker:

<https://www.youtube.com/watch?v=Bz2JYxbkmuY>

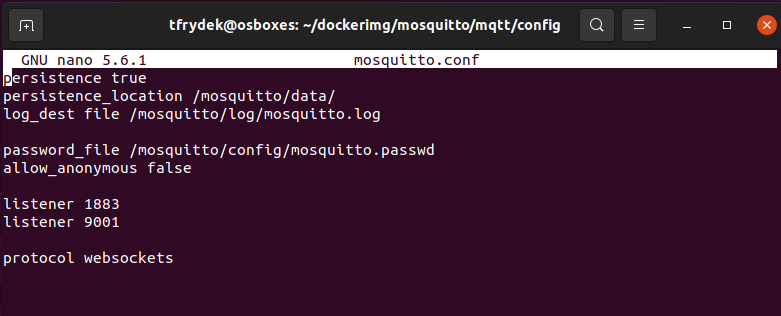
sudo docker run -itd --name=MQTTbroker\_oi --restart=always -p 1883:1883 -p 9001:9001 \

-v /home/tfrydek/dockerimg/mosquitto/mqtt/config:/mosquitto/config \

-v /home/tfrydek/dockerimg/mosquitto/mqtt/data:/mosquitto/data \

-v /home/tfrydek/dockerimg/mosquitto/mqtt/log:/mosquitto/log eclipse-mosquitto:1.6

Create file mosquitto.conf in *config* directory ( /home/tfrydek/dockerimg/mosquitto/mqtt/config ):



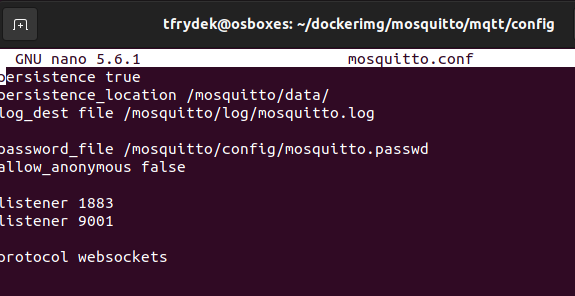
add a user to a new password file. In that case username is *mqttbroker*:

mosquitto\_passwd -c /mosquitto/config/mosquitto.passwd **mqttbroker**

define password

exit

open file mosquitto.conf:



UserName: mqttbroker

UserPassword: LoopEdge.123

Mqtt broker Internal container IP address: 172.17.0.2

InfluxdB internal container IP address: 172.17.0.3

After each modification of container: docker restart ***container\_id***

## **Docker system commands:**

check container is in run:

docker ps

open the container shell:

docker exec -it ***Number of ID container*** sh

Show docker disk usage:

docker system df

A screenshot of a computer

Description automatically generated with medium confidence

Show docker disk usage/detailed information on space usage:

docker system df -v

Text

Description automatically generated

## **InfluxDB:**

192.168.1.169

TomF Internet1!

admin LoopEdge.123

**sudo docker run -itd --restart=always -p 8086:8086 --name=influxdb\_oi -v influxdb:/var/lib/influxdb influxdb:2.2**

sudo docker run -itd --restart=always --net PPS\_Network --ip 172.10.0.3 -p 8086:8086 --name=influxdb -v influxdb:/var/lib/influxdb influxdb:2.2

sudo docker exec -it *influxdb\_oi* /bin/bash // *enter the bash inside influxdb container*

influx bucket create -n PLPPS001\_PROCESS\_DB\_ASSEMBLER -o "mmm" -t “token name” // can also create the bucket in UI

create another influxdb user:

**influx user create -n admin -p LoopEdge.123 -o “mmm” -t “*here\_token\_number” (can be found on UI)***

give ALL write/read access for user admin, use token from owner user:

influx auth create --all-access -o "mmm" -u admin -t “token”

influx -host ip\_addres -port 8086 -username admin -password adm //*log on into influxdb*

*when logged try these commands:*

show databases

create database PLPPS001\_PROCESS\_DB\_ASSEMBLER

select \* from BT1\_DevPoint //*as soon as PLPPS…Process.. is created and data are recorded*

use PLPPS001\_PROCESS\_DB\_ASSEMBLER // select database for use

show measurements // display tables inside used database

save InfluxDB data in csv format:

**influx -database 'database\_name' -execute "SELECT \* FROM table\_name" -format csv > test.csv**

delete specified measurement in InfluxDB:

**influx delete --bucket 'PLPPS001\_ANOMALY\_DB\_ASSEMBLER\_TEST' --start '1970-01-01T00:00:00Z' --stop '2022-08-03T00:00:00Z' --predicate '\_measurement="Assembler\_No1\_10N3\_RetCmd\_T"' -t '*token\_id*' --org 'mmm'**

list all buckets in organisation:

**influx bucket list --org 'mmm' -t '*token\_id*'**

[Back up a specific bucket to a current directory](https://docs.influxdata.com/influxdb/v2.2/reference/cli/influx/backup/#back-up-a-specific-bucket-to-a-directory):

**influx backup --bucket PLPPS001\_PROCESS\_DB ./ --org mmm -t '*token\_id*'**

[Back up all data to a directory](https://docs.influxdata.com/influxdb/v2.2/reference/cli/influx/backup/#back-up-all-data-to-a-directory):

**influx backup /path/to/backup/dir/**

copy folder */influx\_backup* from container into local folder /home/tfrydek..

**docker cp *container\_id*:/influx\_backup /home/tfrydek/dockerimg/TESTS**

copy ALL files from host folder influx\_backup into container folder influx\_backup:

***docker cp /home/tfrydek/dockerimg/TESTS/influx\_backup/. container\_id:/influx\_backup/***

restore bucket PLPPS001 from location /influx\_backup as new bucket names PLPPS002:

***influx restore /influx\_backup --bucket PLPPS001\_PROCESS\_DB --new-bucket PLPPS002\_PROCESS\_DB --org mmm -t 'token\_id'***

restore bucket PLPPS001 from location /influx\_backup. In this case, in InfluxDB the PLPPS001 should NOT(!) exist before restoring action.

***influx restore /influx\_backup --bucket PLPPS001\_PROCESS\_DB --org mmm -t 'token\_id'***

disk usage of influxdb (inside container):

***du -sh /var/lib/influxdb2/***

IMPORTANT LINKS:

Setup Influxdb and Grafana on CentOS:

<https://www.youtube.com/watch?v=xiV4MhL8bOQ>

InfluxDB: Overview, key concepts and demo

<https://www.youtube.com/watch?v=gb6AiqCJqP0>

## **GRAFANA**:

docker run -d -p 3001:3000 grafana/grafana-enterprise

Check usage of port 3000

netstat -pna | grep 3000

OR

ss -ln src :3000

*# create a persistent volume for your data in /var/lib/grafana (database and plugins)*

docker volume create grafana-storage

docker run -d -p 3000:3000 --name=grafana -v grafana-storage:/var/lib/grafana grafana/grafana-enterprise

## **Docker Compose**

Download

1. mkdir -p ~/.docker/cli-plugins/
2. curl -SL https://github.com/docker/compose/releases/download/v2.3.3/docker-compose-linux-x86\_64 -o ~/.docker/cli-plugins/docker-compose

Next, set the correct permissions so that the docker compose command is executable:

1. chmod +x ~/.docker/cli-plugins/docker-compose

Check version:

1. docker compose version

<https://faun.pub/run-multi-container-docker-applications-with-a-single-command-795e1949f379>

Don’t start the services after creating them.

docker-compose --verbose up --no-start

To restart all stopped(created) containers:

sudo docker restart $(docker ps -a -q)

To stop all containers with filtering condition:

docker stop $(docker ps -a -f "label=3M.container.workcenter=PPS\_L1")

To dispay containers with filtering option:

docker ps -f “status=running” created, restarting, running, removing, paused, exited, or dead

Display containers with filter on labels:

docker ps -a -f "label=3M.container.workcenter=PPS\_L1"

start defined docker compose file in detached mode:

docker-compose -f docker-compose.main.yml --verbose up -d

## **Networking:**

docker network ls - list of available networks

Create manually network:

docker network create --driver=bridge --subnet=172.10.0.0/16 --gateway=172.10.0.1 IOT\_Network

Create manually influxdb container :

sudo docker run -itd --restart=always --net IOT\_Network --ip 172.10.0.3 -p 8086:8086 --name=influxdb -v influxdb:/var/lib/influxdb influxdb:2.2

Connect a container to a network:

docker network connect IOT\_Network\_PSD\_WB1 MQTT\_Broker

docker network connect IOT\_Network\_PSD\_WB1 InfluxDB

The **IMAGE\_NAME** can be one of these values:

- The fully qualified image name, for example, “ghcr.io/baeldung/my-application:1.2.3”

- The image hash generated by Docker, for example, “c85146bafb83”

## **Docker image handling:**

Save docker image as file:

docker image save -o file\_name.tar IMAGE\_NAME

Import docker image from file:

docker image import file\_name.tar IMAGE\_NAME:IMAGE\_TAG

docker load -i file\_name.tar

## **Telegraf:**

Pull telegraf image:

docker pull telegraf

docker run --network=IOT\_Network --ip=172.10.0.200 -d -P --name=telegraf telegraf

Docker daemon permissions:

Typically, telegraf must be given permission to access the docker daemon unix socket when using the default endpoint. This can be done by adding the telegraf unix user (created when installing a Telegraf package) to the docker unix group with the following command:

sudo usermod -aG docker telegraf

If telegraf is run within a container, the unix socket will need to be exposed within the telegraf container. This can be done in the docker CLI by add the option -v /var/run/docker.sock:/var/run/docker.sock or adding the following lines to the telegraf container definition in a docker compose file:

volumes:

- /var/run/docker.sock:/var/run/docker.sock

docker run --network=IOT\_Network --ip=172.10.0.200 -d -P -v /var/run/docker.sock:/var/run/docker.sock --name=telegraf telegraf

login to the telegraf container:

docker exec -it *container\_id* /bin/bash

print environment variables from container:

docker exec *container*\_id printenv

Docker clean up of /var/lib/docker/overlay:

docker system prune --all --volumes –force

Remove all log file:

find /var/lib/docker/containers/ -type f -name “\*.log” -delete

after that restart all containers