Docker Compose

Link script: https://krspiced.pythonanywhere.com/chapters/project_pipeline/docker/compose.html

Link teaching guide:http://krspiced.pythonanywhere.com/teaching_guide/07_docker_pipeline/docker_compose/

Goal

Students set up a simple pipeline with several docker containers

!! IF the already had the MongoDB encounter: you can do the warmup+example with mongodb instead of postgres

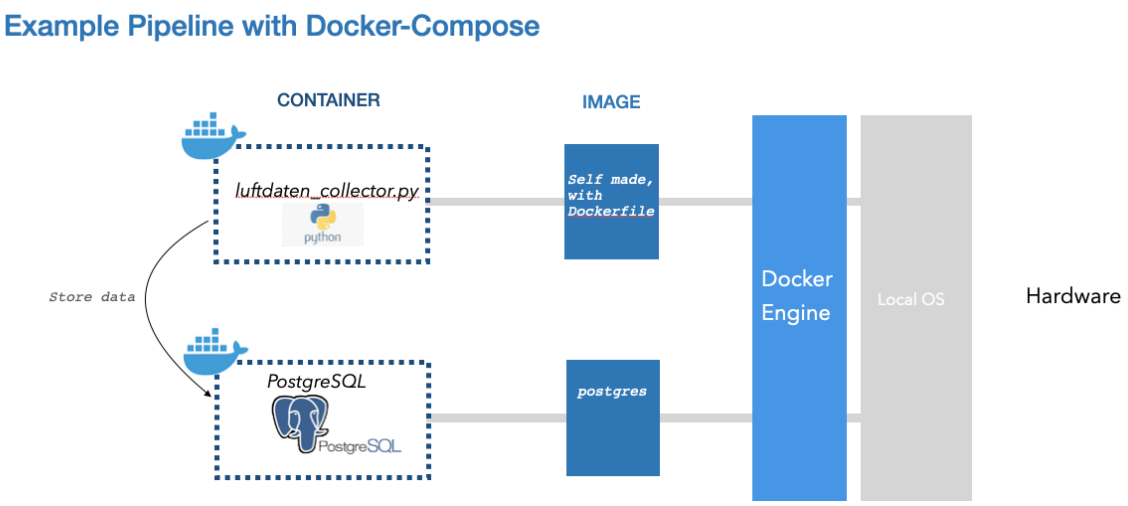
Warm up

Postgres inside Docker

```
docker run -d --name mypg -p 5555:5432 -e POSTGRES_PASSWORD=xxxx postgres

### Connect from your own computer

psql -h 127.0.0.1 -p 5555 -U postgres

WINDOWS:
psql -h 192.168.99.100 -p 5555 -U postgres

###Next, connect from inside Docker:

docker exec -it mypg psql -p 5432 -U postgres
```

Outline of the encounter:

- **Docker Compose**: define all individual services in a single file.
  - **yml file**
- Example: take our data from luftdaten and store them in postgres
- **Our Docker pipeline**: idea each service is a Docker container



Docker compose

How does it work?

Docker compose **starts with a file, a yml**, where you define what your pipeline or architecture consists for. With smallest unit being a container. We need to think about 3 things:

- ports
- environment variables
- shared volumes

Docker compose file **has paragraphs, each of which defines a container.** The indentation is important.

yml: yet another markup language, some kind of engineering joke

```
version: '3'
services:
  ludtdaten_collector:
    build: luftdaten/
    volumes:
    - ./luftdaten/:/app
    links:
    - mypg
  mypg:
    image: postgres
    ports:
    - 5555:5432
    environment:
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=1234
```

`volumes`: take what is inside the specified folder in my computer and copy it to the folder code in the container

`link`: list to the other containers you want to communicate

—

- `mkdir` luftdaten
- inside luftdaten

```
# Use an official Python runtime as a parent image
FROM python:3.6-slim

# Set the working directory to /app
WORKDIR /app

# Copy the requirements file into the container at /app
# (a text files with all the libraries you want to install)
COPY requirements.txt /app
COPY get_luftdaten.py /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Run app.py when the container launches
CMD ["python", "get_luftdaten.py"]
```

get_luftdaten.py

```


import requests
import json
import random
import time
import logging

SENSOR_URL = "http://api.luftdaten.info/static/v1/sensor/{}/"

def pick_luftdaten_values(sensor_id):
    # Sensordaten für Luftdaten (p1/p2=Feinstaubwerte, DHT für Temp und Luftfeuchte) abfragen
    # dazu die api von luftdaten.info nutzen
    result = requests.get(SENSOR_URL.format(sensor_id))
    data = result.json()
    time = data[0]['timestamp']
    PM25 = data[0]['sensordatavalues'][0]['value']
    PM10 = data[0]['sensordatavalues'][1]['value']
    lon = data[0]['location']['longitude']
    lat = data[0]['location']['latitude']
    return time, PM25, PM10, lon, lat

sensors_stuttgart = [1625, 2023, 7037]

while True:
    sensor_id = random.choice(sensors_stuttgart)
```

```
    t_stamp, PM25, PM10, lon, lat  = pick_luftdaten_values(1625)
    #print(f'sensor data for sensor {sensor_id}: PM 2.5 {PM25}, PM 10 {PM10}, time {t_stamp}')
    logging.critical(f'sensor data for sensor {sensor_id}: PM 2.5 {PM25}, PM 10 {PM10}, time {t_stamp}')
    time.sleep(30)
```

- requirements.txt

```
requests

random
time
logging
```

Let's add a second container with a `postgres` image:

```
mypg:
    image: postgres
    ports:
    - 5555:5432
    environment:
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=1234
```

Build a docker image for the pipeline: (every time you change the code)

```
docker -compose build
```

Run the containers from the image

```
docker -compose up
```

Add Postgres Stuff into Script

```


from sqlalchemy import create_engine


engine = create_engine('postgres://postgres:1234@mypg:5432/postgres')

create_query = '''
CREATE TABLE IF NOT EXISTS luftdaten (
timestamp DATE,
pm25 REAL,
```

```
pm10 REAL,
lon TEXT,
lat TEXT);
'''
engine.execute(create_query)

sensors_stuttgart = [1625, 2023, 7037]

while True:
    sensor_id = random.choice(sensors_stuttgart)
    t_stamp, PM25, PM10, lon, lat  = pick_luftdaten_values(1625)
    logging.critical(f'sensor data for sensor {sensor_id}: PM 2.5 {PM25}, PM 10 {PM10}, time {t_stamp}')
    engine.execute(f"""INSERT INTO luftdaten VALUES ('{t_stamp}','{PM25}','{PM10}','{lon}','{lat}');""")
    logging.critical('----INSERTED INTO POSTGRES!------\n')
    time.sleep(30)
```

Check that data is in database

```
docker start docker_compose_mypg_1
```

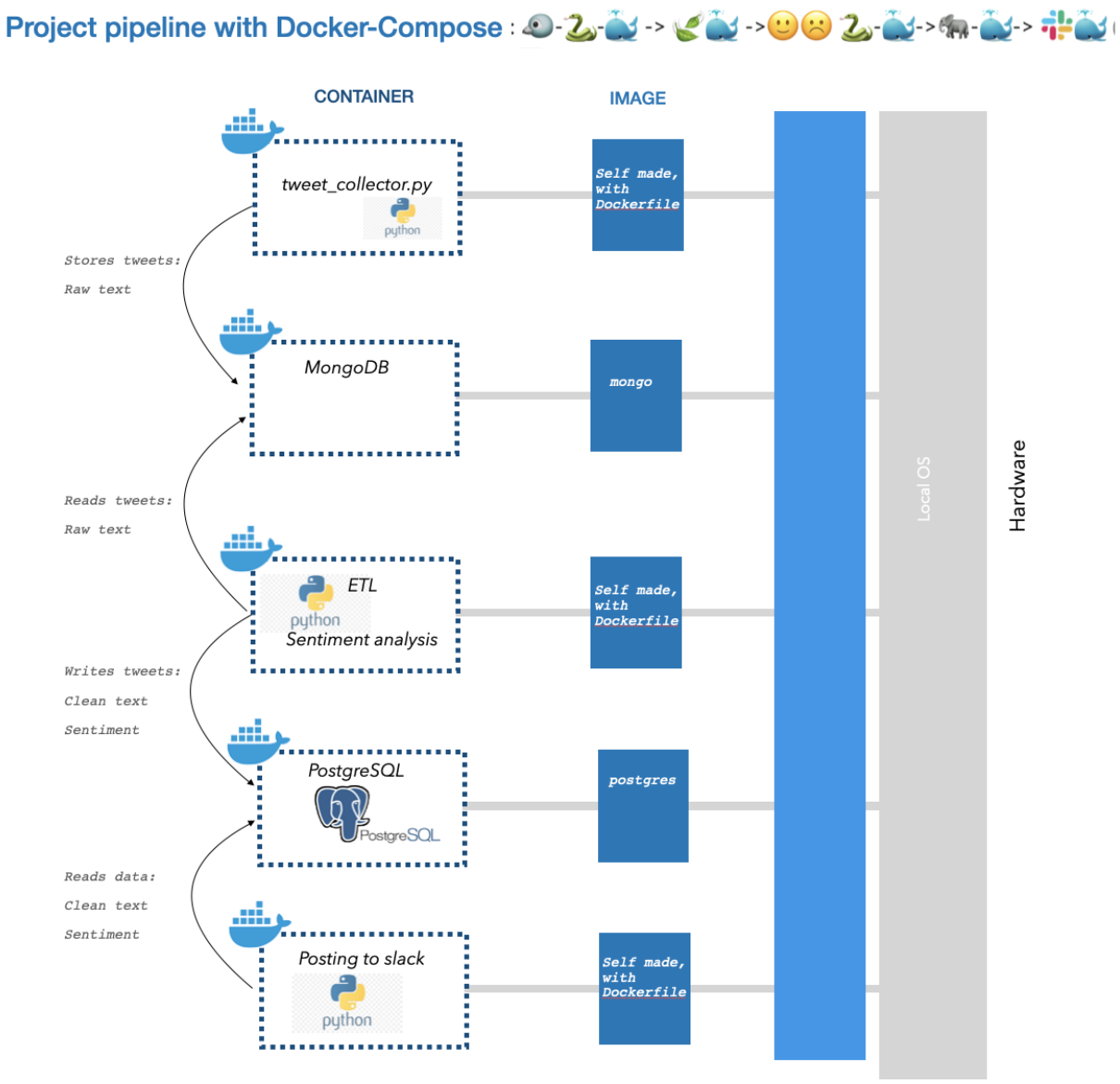
---

in a separate terminal:

```
psql -h 127.0.0.1 -p 5555 -U postgres
```

Wrap up: Docker pipeline for the week

What are the elements of the pipeline of this week? For each step we want an individual Docker container. To define and run all together so that they speak to each other we need one script, this script is Docker compose.



Known questions

Issue**: Linux users have to install docker-compose separately**

A: Point them to do it ;)

Useful links

- ...
-

TO DOs