1. Устанавливаем докер согласно инструкциям
2. Создаем сеть внутри докера

Изображение выглядит как текст, снимок экрана, Шрифт

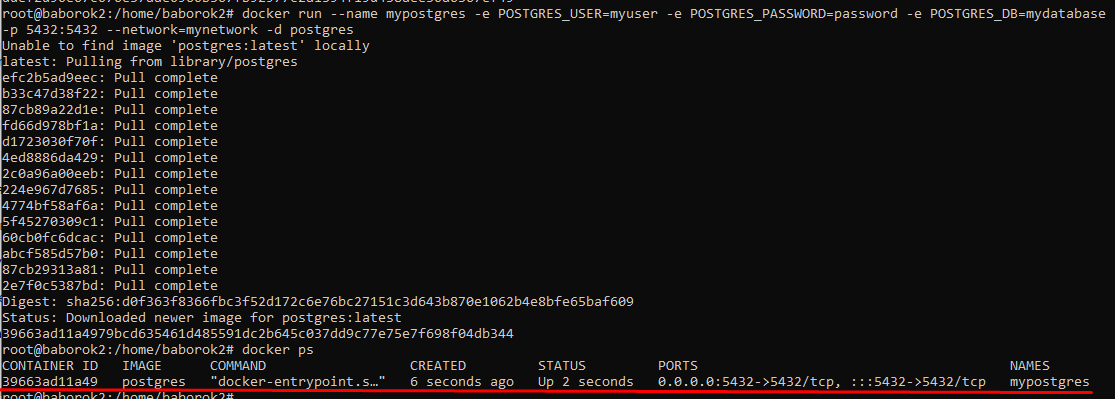
Автоматически созданное описание

1. Следующей командой подтягиваем Postgres

docker run --name mypostgres -e POSTGRES\_USER=myuser -e POSTGRES\_PASSWORD=password -e POSTGRES\_DB=mydatabase -p 5432:5432 --network=mynetwork -d postgres

Использованные флаги позволяют задать базовую конфигурацию для юзера дб пароля и прочего

Далее командой docker ps проверяем работает ли контейнер



Пробросили нужные порты (не забыв открыть их на исходной машине тоже для доступа допустим извне для инструментов DBeaver

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Ввели соотвественно поля юзер датабейз и прочее из команды выше, видим все что представлено на нашей машине

Но настраивать привычнее через CLI, поэтому так и сделаем

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание  
Добавили юзера ньююзер и добавили ему все права на mydatabase на этом настройку, наверное, можно завершить. Приступим к разработке python скрипта

Находим откуда можно доставать данные в интернете  
https://open-meteo.com/  
Ресурс подходит +- под наши требования  
Будем курлить с их примера

$ curl ["https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&current=temperature\_2m,wind\_speed\_10m&hourly=temperature\_2m,relative\_humidity\_2m,wind\_speed\_10m"](https://api.open-meteo.com/v1/forecast?latitude=52.52&longitude=13.41&current=temperature_2m,wind_speed_10m&hourly=temperature_2m,relative_humidity_2m,wind_speed_10m)

{

"current": {

"time": "2022-01-01T15:00"

"temperature\_2m": 2.4,

"wind\_speed\_10m": 11.9,

},

"hourly": {

"time": ["2022-07-01T00:00","2022-07-01T01:00", ...]

"wind\_speed\_10m": [3.16,3.02,3.3,3.14,3.2,2.95, ...],

"temperature\_2m": [13.7,13.3,12.8,12.3,11.8, ...],

"relative\_humidity\_2m": [82,83,86,85,88,88,84,76, ...],

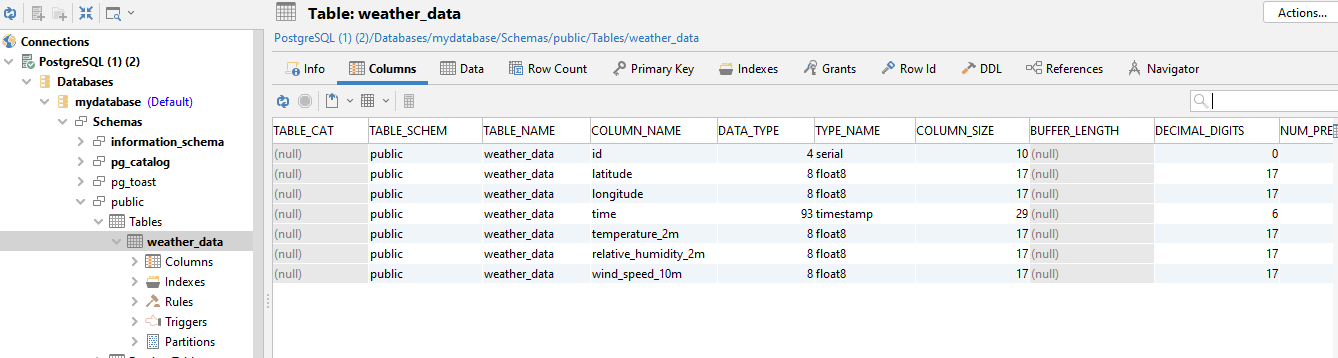
}

}

Отлично, данные есть, можно создавать схему в БД  
Для практики создадим через UI инструмент

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Отлично, видим, что получилось  
  
Теперь напишем пробный python скрипт, чтобы подтягивать данные оттуда и класть их в нашу базу

Чтобы было проще все это потом засунуть в докер создадим env в котором будем разрабатывать питон скрипт

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Внутри него скачали 3 пакета

psycopg2 – для коннекта к постгресу

schedule – чтобы контролировать время (можно через слип но не обязательно)

requests – чтобы курлить эффективно нашу апишку

Со всем скриптом можно ознакомиться в самой репе.

Далее пишем Докерфайл чтобы поднимать наш сервис

FROM python:3.9-slim – подтягиваем питон

WORKDIR /app

COPY requirements.txt /app/ - подтягиваем требования в вольюм контейнера

RUN pip install --no-cache-dir -r requirements.txt – устанавливаем

COPY . /app/

ENTRYPOINT ["python", "weather\_collector.py"] – запускаем скрипт

Можно было бы в теории подтянуть целую ось, например alpine, но делаем ставку, что установка уже будет проходить в UNIX системах

Далее пишем docker-compose файл, чтобы поднимать все сервисы разом

version: '3.8'

services:

db:

image: postgres:13

container\_name: mypostgres

environment:

POSTGRES\_USER: myuser

POSTGRES\_PASSWORD: password

POSTGRES\_DB: mydatabase

ports:

- "5432:5432" # Проброс порта

healthcheck:

test: ["CMD-SHELL", "pg\_isready -U myuser -d mydatabase"] – хелс чек чтобы проверить что база поднялась

interval: 10s

retries: 5

start\_period: 20s

timeout: 5s

weather\_collector:

build:

context: . # Path to your Dockerfile

dockerfile: Dockerfile

depends\_on:

db:

condition: service\_healthy – зависимость что поднялась база

environment:

DATABASE\_HOST: mypostgres

DATABASE\_PORT: 5432

DATABASE\_NAME: mydatabase

DATABASE\_USER: myuser

DATABASE\_PASSWORD: password

Вот и все установка всего добра максимально простая

Скопировать репу git clone

В директории сделать docker compose up –build

Общие мысли по поводу того\, что можно было бы улучшить и сделать по-другому:

Разнести процесс подтягивания данных и загрузки в разные сервисы.

Добавить брокер сообщений, если источников было бы несколько, например сделать кафку(для разных источников), для сохранения целостности и хронологии сообщений, (условие актуально, если бы поток данных был более объемный)