

"Data Engineering Zoom Camp"

- week 1 : introduction & prerequisites
 - course overview
 - Introduction to GCP
 - Docker
- week 2 : Data ingestion
 - Data lake
 - workflow orchestration "AirFlow"
- week 3 : Data Warehouse
 - Big Query
 - partitioning and clustering "BQ"
- week 4 : Analytics engineering
 - dbt (data build tool) "DBT"
 - Big Query & dbt
- week 5 : Batch processing
 - spark
 - spark sql "spark"
- week 6 : streaming
 - kafka
 - schemas (avro) "Kafka"
- week 7, 8 & 9 : project
 - putting what you learn in practice.



Technologies:

- Google Cloud platform (GCP)
 - cloud based auto scaling platform
 - GCS [Google Cloud Storage]
 - Big Query [Data warehouse]
- Terraform
 - infrastructure as code (IaC)
- Docker
 - Contain erization
- SQL
 - Data Analysis & Exploration
- Airflow
 - [pipeline orchestration]
- dbt
 - Data Transformation
- Spark
 - Distributed processing
- kafka
 - streaming



1.2.1 "introduction to docker"

postgres : Free and open source relational database management system.

- kubernetes: Open source Container orchestration (تنسيق)
- AWS batch system for automating software deployment.

Docker : platform as a service product uses OS-level virtualization to deliver software in packages called containers.

- containers are "isolated" from each others.
- They can communicate with each other through channels.

- why we should use docker ?

- local experiments
- Integration tests (CI/CD)
- Re producibility
- Running pipelines on cloud (AWS Batch, kubernetes jobs)
- Spark
- serverless (AWS lambda, Google function)



we will work with git bash

- > `mkdir docker` # create directory called docker
- > `code .` # open visual studio with directory
- > `docker run hello-world`
to test that docker is working
- > `docker run -it ubuntu bash`
run in interactive mode
name of img → ubuntu
command to execute in img → bash

we will do some magic [we will remove all folders from our img]

- > `rm -rf / --no-preserve-root`
Remove all files.

- > `docker run -it --entrypoint=bash python`
what to execute when run → bash
img name → python

Some times when running docker from [git bash] we may have error of winpty builtin method to deal with -it interactive mode.

solution winpty docker run -it ubuntu bash
put winpty on first of docker every time

or `echo "alias docker='winpty docker'" >> ~/.bashrc`

> docker run -it --entrypoint = bash python:3.9

entrypoint what to execute 3.9 img

> pip install pandas
to install pandas inside this img.

> pandas. -- version --
to get version

Making a Docker File :

• inside Dockerfile "visual studio"

```
> FROM python: 3-9
> RUN pip install pandas
> ENTRYPOINT ["bash"]
```

- that would use python:3.9, img and overwrite it to install pandas and save new img and make the endpoint bash

> docker build -t test: pandas → looking for docker file in pwd

- we now have next img called test:pointers
- with - pandas installed
- direct endpoint bash



calling this new img:

```
> docker run -it test:pandas
>>> #bash
>>> python
>>> import pandas as pd
↳ Now we can import pandas without any problems.
```

Running python file:

- we will make another file "-.py" in the same folder.
- let's call it pipeline.py

```
import sys
import pandas as pd
print (sys.argv) → print argument what I in while container
day = sys.argv[1] → day is first arg
print ('Job finished successfully for day = {day}')
```

- we will change the docker file

FROM python:3.9

RUN pip install pandas

WORKDIR

app → make directory called app

COPY

pipeline.py

pipeline.py

→ copy this file to pwd in the img

ENTRYPOINT ["python", "pipeline.py"]

↳ make entr python & run →



> docker run -it test:pandas 2021-01-15
First argument
Entry point
\$ ['Pipeline.py', '2021-01-15']
\$ Job Finished for day= 2021-01-15
argument [1]
give this argument
to whatever run
in docker.

Video 2 Ingesting NY Taxi Data to postgres

• we will run postgres "relational Database"
to investigate NY Database.

```
> docker run -it \-e POSTGRES-USER="root" \
-e POSTGRES-PASSWORD="root" \-e POSTGRES-DB="ny-taxi" \
-v my-directory : /var/lib/postgresql/data \
-p 5432:5432 \
postgres:13
```

run image and pass
credentials to it
Mounting directory outside img to img
port
docker img name
postgres
database
server

> docker ps # to see and explore
current active docker imgs.

Now we have SQL data base running from
Container, so How to connect to it.

we will use pgcli "Postgres cli"

pgcli -h localhost -p 5432 -u root -d ny-taxi

call
this server
from another
command

> \.at explore data base
> SELECT 1 ;

Explore dataset

[linux command]

> less yellow-trip.csv
↳ to see some info about data.
> explore.
↳ to open file explorer

> head -n 100 yellow-trip.csv > yellow.csv
↳ save first 100 row to another file.

> wc -l yellow.csv
↳ word count line

Make connection from python to postgres

connect python to postgres From SQLAlchemy import create_engine
engine = create_engine('postgres://root:root@localhost:5432/ny-taxi')
print(pd.io.sql.get_schema(df, name='yellow-taxi-data', con=engine))
get the schema what to write to make database using postgres
connection type

CREATE TABLE yellow-taxi-data (
vendorID BIGINT,
trip-distance FLOAT(53)



Data set is big, so we will use chunk read data on chunks.

~~df = pd.read_csv('yellow-taxi.csv')~~
Just an iterator
df = pd.read_csv('yellow-taxi.csv', iterator=True, chunksize=10000)

df = next(df-iter)
↳ next chunk

while True:

Appending data to SQL using Python
df = next(df-iter)
df.to_sql(name='yellow-taxi', con='engine', if_exists='append')
↳ Append chunk to the database

> SELECT count(*) FROM yellow-taxi-data;

DE Zoom Camp 1-2-3 Connecting pgadmin and postgres

SELECT max(column-name), min(—), max(—)
FROM yellow-taxi;

pg Admin

web-based GUI tool to interact with postgres database session.

docker run -it \

-e PGADMIN-DEFAULT-EMAIL="admin@admin.com" \

-e PGADMIN-DEFAULT-PASSWORD="root" \


-p 8080:80 \

dfage pgadmin4 → img name



Connect 2 containers

- > docker network creat pg-network
creat network with this name to enable
other imgs to see each others.

we will edit in code of  postgres database
pgadmin

- > docker run -it \
--~~network~~ = pg-network
postgres:13

- > docker run -it \
--network = pg-network
postgres:13

1.2.4 Dockerizing the ingestion script

- > jupyter nbconvert --to=script upload-data.ipynb
to convert code to script

- > Docker ps
give you list of IDs of active imgs

- > Docker kill ID

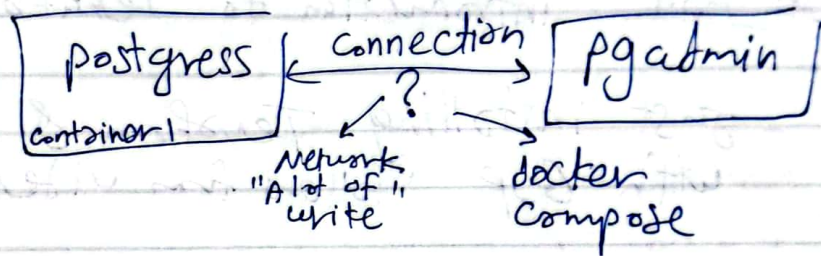


1.2.5 Running postgres and pgAdmin in docker compose

In previous work we could establish network between 2 containers.

we will use another method here docker compose

Target



we write code of 2 containers:

docker-compose.yml

services:

> pgdatabase:

image: postgres:13

environment:

- postgresql-USER=root

⋮

volumes:

- './ny-taxi' : volume mapping

ports:

- '5432:5432'

> pgadmin:

image: dpage/pgadmin4

⋮

> docker-compose

up

to run it


>

//

down



1.2.6 SQL Refresher

- some play with SQL you need  do it by your hands.
- check Alvaro Navas notes

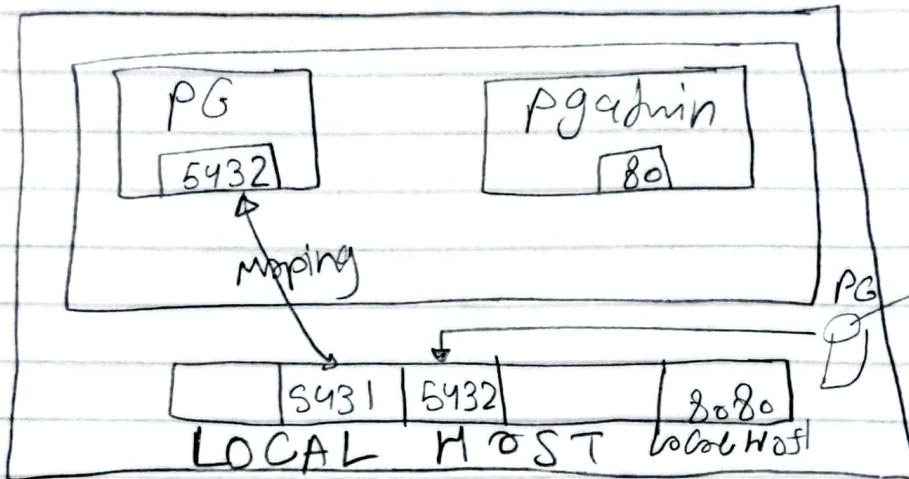
DE 1.3.1 introduction to Terraform & GCP

- just installing Terraform & connecting with GCP "follow from video"
- Terraform IAC "Infrastructure AS a Code" that allows you to connect to GCP "Google Cloud Platform" and even more like "AWS, Alibaba..."
 - All of that using CLI From same code area same place to connect AWS, GCP....
 - Allows you to control infrastructure and versions.



1.4.2 Port Mapping (Bonus)

- we use port mapping as some time we may have other app on our local machine that use same port.



- setting up environment on cloud
"run all of work from docker to compose postgres on a virtual Machine on Gcp"

