**WEEK-1 Basics and Setup**

**Google Cloud Services**

****

**Installing Python on Ubuntu:**

<https://phoenixnap.com/kb/how-to-install-python-3-ubuntu>

Step 1: Update and Refresh Repository Lists

Open a terminal window, and enter the following:

**sudo apt update**

Step 2: Install Supporting Software

The software-properties-common package gives you better control over your package manager by letting you add PPA (Personal Package Archive) repositories. Install the supporting software with the command:

**sudo apt install software-properties-common**

Step 3: Add Deadsnakes PPA

Deadsnakes is a PPA with newer releases than the default Ubuntu repositories. Add the PPA by entering the following:

**sudo add-apt-repository ppa:deadsnakes/ppa**

The system will prompt you to press enter to continue. Do so, and allow it to finish. Refresh the package lists again:

**sudo apt update**

Step 4: Install Python 3

Now you can start the installation of Python 3.8 with the command:

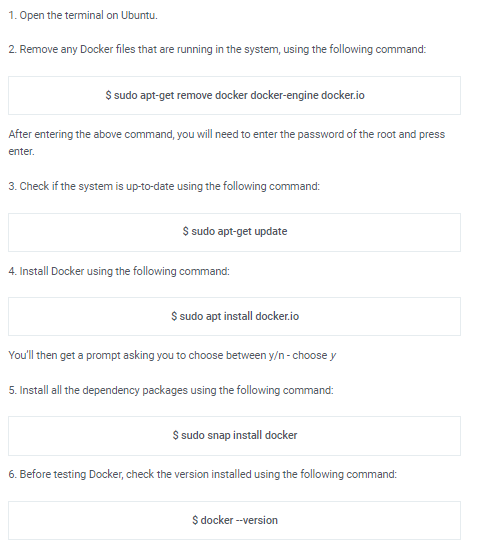
**sudo apt install python3.8**

**Setting Python3 as default compiler**

**sudo update-alternatives --set python /usr/bin/python3.8**

**python** - - **version**

**Installing Docker on ubuntu**

****

**Executing Docker Commands as Non-root user**

**sudo usermod -aG docker $USER**

**docker ps**

**Install Anaconda on Ubuntu**

1- Follow the steps from the blog [**https://phoenixnap.com/kb/how-to-install-anaconda-ubuntu-18-04-or-20-04**](https://phoenixnap.com/kb/how-to-install-anaconda-ubuntu-18-04-or-20-04)

2- To use gui (anaconda navigator)

**export PATH=/home/yourUserName/anaconda3/bin:$PATH**

**anaconda-navigator**

**Installing Sublime text editor**

<https://linuxize.com/post/how-to-install-sublime-text-3-on-ubuntu-20-04/>

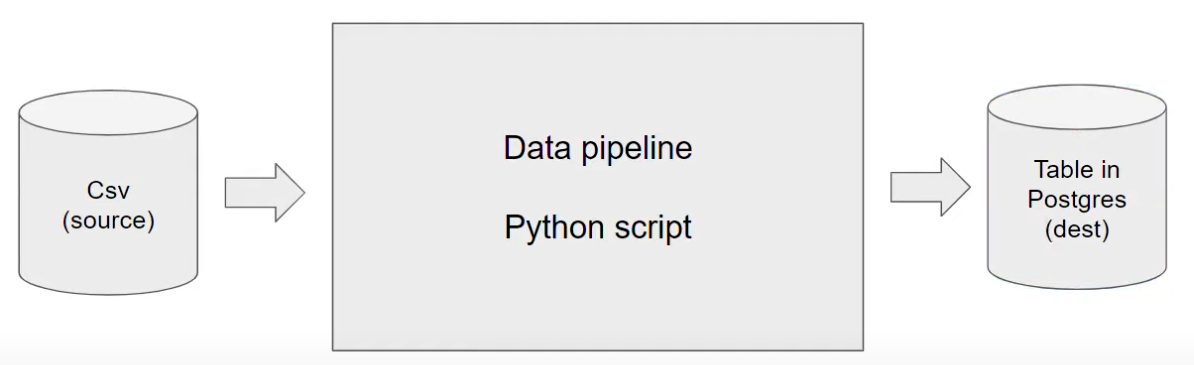
**– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –**

**Data Engineering Basics**

Data engineers are responsible for consuming this data, designing a system that can take this data as input from one or many sources, transform it, and then store it for their customers / end-users to analyze at scale.

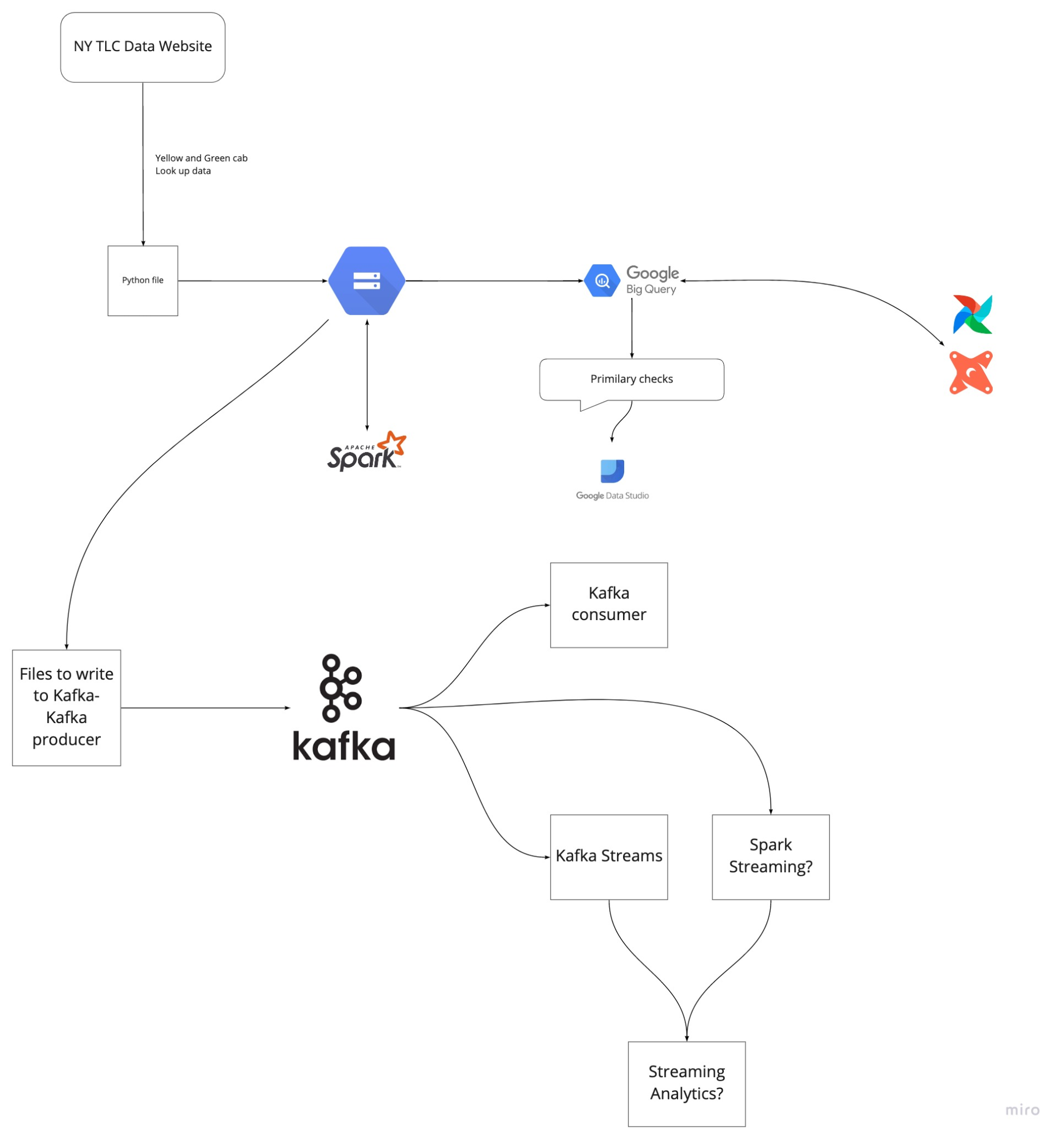
**Data Pipeline**

A data pipeline is a means of moving data from one place (the source) to a destination (such as a data warehouse). Along the way, data is transformed and optimized, arriving in a state that can be analyzed and used to develop business insights.



**Course Outcome:**

Designing an end-to-end data pipeline to automate data flows.



Technologies to be used:

* *Google Cloud Platform (GCP)*: Cloud-based auto-scaling platform by Google
  + *Google Cloud Storage (GCS)*: Data Lake
  + *BigQuery*: Data Warehouse
* *Terraform*: Infrastructure-as-Code (IaC)
* *Docker*: Containerization
* *SQL*: Data Analysis & Exploration
* *Airflow*: Pipeline Orchestration
* *DBT*: Data Transformation
* *Spark*: Distributed Processing
* *Kafka*: Streaming

**2\_Docker\_Sql**

Docker is a container management service. The keywords of Docker are develop, ship and run anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.

* Github link: [data-engineering-zoomcamp/week\_1\_basics\_n\_setup/2\_docker\_sql at main · DataTalksClub/data-engineering-zoomcamp (github.com)](https://github.com/DataTalksClub/data-engineering-zoomcamp/tree/main/week_1_basics_n_setup/2_docker_sql)
* Videos playlist: [Data Engineering Zoomcamp - YouTube](https://www.youtube.com/playlist?list=PL3MmuxUbc_hJed7dXYoJw8DoCuVHhGEQb)

Reference tutorial: [Docker Tutorial: Get Going From Scratch – Stackify](https://stackify.com/docker-tutorial/)

**Running docker images:**

docker run -it ubuntu bash

docker run -it python:3.8

**Installing pandas package in python container:**

docker run -it entrypoint=bash python:3.8

pip install pandas

python

**Example-1 with dockerfile:**

**pipeline.py:**

import pandas as pd

# some transformations

print(“job completed successfully”)

**Dockerfile:** (Please note that the name of the file has to be "Dockerfile" with "D" as capital)

FROM python:3.9

RUN pip install pandas

WORKDIR /app

COPY pipeline.py pipeline.py

ENTRYPOINT [ "bash" ]

**To run dockerfile:**

docker build -t test:pandas .

**Run the image:**

docker run -it test:pandas

**Example-2 with dockerfile:**

**pipeline.py:**

import sys

import pandas as pd

print(sys.argv)

day = sys.argv[1]

# some fancy stuff with pandas

print(f'job finished successfully for day = {day}')

**Dockerfile:**

FROM python:3.9

RUN apt-get install wget

RUN pip install pandas sqlalchemy psycopg2

WORKDIR /app

COPY pipeline.py ingest\_data.py

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

**To run dockerfile:**

docker build -t test:pandas .

**Run the image:**

docker run -it test:pandas 2022-01-18

**Running a PostgreSql on Docker:**

docker run -it \

-e POSTGRES\_USER="root" \

-e POSTGRES\_PASSWORD="root" \

-e POSTGRES\_DB="ny\_taxi" \

-v $(pwd)/ny\_taxi\_postgres\_data:/var/lib/postgresql/data \

-p 5432:5432 \

postgres:13

* postgres container defined above has following:
  + - e for environment variables
  + “ny\_taxi” is the name given to the DB
  + -v for volumes (storage)
  + Format is -v (path in host machine):(path in container)
  + - p for ports in host:container
  + Note: folders in host should be created before running the command

**CLI for PosgtreSql:**

sudo apt-get install libpq-dev python3.8-dev

sudo pip install pgcli

**Connect to DB**

pgcli -h localhost -p 5432 -u root -d ny\_taxi

**Working with Jupyter Notebooks**

* Cheatsheet: [Jupyter\_Notebook\_CheatSheet (edureka.co)](https://www.edureka.co/blog/wp-content/uploads/2018/10/Jupyter_Notebook_CheatSheet_Edureka.pdf)
* Get the source file

wget <https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2021-01.csv>

* More info on fields [data\_dictionary\_trip\_records\_yellow.pdf (nyc.gov)](https://www1.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf)
* We use pandas to import dataset, transform and load to target. More info on pandas can be found at [Python-Pandas-Cheat-Sheet.png (1365×768) (intellipaat.com)](https://intellipaat.com/mediaFiles/2018/12/Python-Pandas-Cheat-Sheet.png)
* Jupyter Notebook used can be found at [git-hub url](https://github.com/FaisalMohd/data-engineering-zoomcamp/blob/main/week_1_basics_n_setup/Practice/upload-data-prac.ipynb)

**Connecting pgAdmin and Postgres with Docker networks**

* PGAdmin is a **web-based GUI tool used to interact with the Postgres database sessions**, both locally and remote servers as well.
* Also used for database administration.

Download pgAdmin docker container at [Download (pgadmin.org)](https://www.pgadmin.org/download/pgadmin-4-container/) or follow instructions to pull the container from docker hub [dpage/pgadmin4 - Docker Image | Docker Hub](https://hub.docker.com/r/dpage/pgadmin4/).

To create a docker virtual network:

docker network create pg-network

To run the pgAdmin:

docker run -it \

-e PGADMIN\_DEFAULT\_EMAIL="admin@admin.com" \

-e PGADMIN\_DEFAULT\_PASSWORD="root" \

-p 8080:80 \

--network=pg-network \

--name pgadmin \

dpage/pgadmin4

To login:

Go to localhost:8080 >> enter email and password

**Note:**

PostgresSql DB and pgAdmin are both running in different docker containers. So we need a virtual docker network to connect these client-server apps.

To run the container in virtual network:

docker run -it \

-e POSTGRES\_USER="root" \

-e POSTGRES\_PASSWORD="root" \

-e POSTGRES\_DB="ny\_taxi" \

-v $(pwd)/ny\_taxi\_postgres\_data:/var/lib/postgresql/data \

-p 5432:5432 \

--network=pg-network \

--name pg-database \

postgres:13

Now to login and connect to DB:

Go to localhost:8080 >> enter email and password >> Give name to this connection >> Enter Server Credentials (Host name / Address), (Username, Password, Port) >> Use query editor to run queries.

**Docker Compose:**

* Instead of running pgAdmin and Postgresql engine from separate terminals, commands;; we use “docker compose” to have them running in one command.
* **Docker Compose** is a tool for running multi-container applications on Docker defined using the Compose file format.

**Dockerizing the Ingestion script:**

To convert the .ipynb notebook to .py file

jupyter nbconvert --to=script upload-data-prac.ipynb

Note: drop table yellow\_taxi\_data; before running ingestion script as we recreate it in script.

Few key-points in [ingest\_data-prac.py](https://github.com/FaisalMohd/data-engineering-zoomcamp/blob/main/week_1_basics_n_setup/Practice/ingest_data-prac.py) file:

* The standard Python library **argparse** is used to incorporate the parsing of command line arguments. More on this library [here](https://towardsdatascience.com/a-simple-guide-to-command-line-arguments-with-argparse-6824c30ab1c3)

Run the script:

- If there is no “pandas” library installed locally, run the command with the prefix ‘!’ on Anaconda Jupyter Notebook or install pandas locally.

python ingest\_data-prac.py \

--user=root \

--password=root \

--host=localhost \

--port=5432 \

--db=ny\_taxi \

--table\_name=yellow\_taxi\_trips \

--url="https://s3.amazonaws.com/nyc-tlc/trip+data/yellow\_tripdata\_2021-01.csv"

To Dockerize the script (modify Dockerfile):

FROM python:3.9.1

# We need to install wget to download the csv file from S3 bucket

RUN apt-get install wget

# psycopg2 is a postgres db adapter used by sqlalchemy for python

RUN pip install pandas sqlalchemy psycopg2

WORKDIR /app

COPY ingest\_data-prac.py ingest\_data.py

ENTRYPOINT [ "python", "ingest\_data.py" ]

Build the image and Run it:

docker build -t taxi\_ingest:v001 .

docker run -it \

--network=pg-network \

taxi\_ingest:v001 \

--user=root \

--password=root \

--host=pg-database \

--port=5432 \

--db=ny\_taxi \

--table\_name=yellow\_taxi\_trips \

--url="<https://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2021-01.csv>"

Note:

* It's optional to drop the table before running the container as the script will automatically overwrite/replace it.
* Before the name of the image, we specify the name of the network.
* As we have pgAdmin already on same network, we can connect to DB from container
* In the arguments here, we are pointing to   
   (hostname, db\_name, tbl\_name = pg\_database, ny\_taxi, yellow\_taxi\_trips)

Useful docker commands:

* To stop all running containers: docker kill $(docker ps -q)
* To remove all containers: docker rm $(docker ps -a -q)
* To remove all images of containers: docker rmi $(docker images -q)

**Running Postgres and pgAdmin with Docker Compose:**