Lab Assigment 1

DS 6600: Data Engineering 1

Andres Castellanos

Problem 1:

I created a new GitHub repository in which the URL is below:

https://github.com/CastelAndres/DS6600_lab1.git

Problem 2:

Describe, in words, whether a virtual machine, a container, a virtual environment, or the global environment of a single computer is best suited for each of the following situations. Be clear about why the option you choose works best, and also about why the other options are insufficient or are overkill.

- a. For the scenario of Meals on Wheels a Container is the best option. I recall that a great benefit of using containers is that it will let you "package" the web portal and mobile app so that it runs identical for ANY chapter, regardless of hardware. This also allows the current data base to be connected to each new container for each new chapter created, which containers are also easily scalable. A virtual machine wont be the best option as it provides isolation, which means that duplicating full OS images for EVERY new chapter will come with a price and will be resource intensive. A virtual environment will only isolate Python packages, which is not sufficient to manage their apps,databases, and website. Lastly, the global environment would basically be stuck sharing one system. This would be a nightmare to manage as it would be hard to separate volunteer and client data securely, and scaling would be really hard.

- b. In this scenario the Legal Aid Justice Center would be better off using a virtual environment as they're dealing with a data cleaning task. A Python virtual environment such as pandas, will pretty much allow you to clean the data. This virtual environment also allows for isolation of the project dependencies. Another great perk as to why its a great option is that once you have cleaned the data, you can easily export what you need and wont have to worry about hosting/deployment. A container in this situation is possible, but overkill, since the task requires just a one-time report you wont have to worry about a reproducible service/app. A virtual machine would just be a waste of time and resources to run a whole extra OS to

process just this single task. A Global environment would work as well, but the issue become when your system Python environment gets cluttered with various analysis libraries. The virtual environment keeps things cleaner/easier to reproduce if you need to do so later.

- c. For this task it appears that a container would be best suited. Since we need Ubuntu system packages the container would let us build an image that installs the necessary packages and our Python environment as well. This makes it so we are not forcing others to run full Ubunty VM's or change their OS. This also allows for distribution options and allows us to have controlled dependencies. A virtual machine could run Ubunty, but it is slower to start and makes it harder to distribute when compared to containers. A virtual environment is only able to manage Python libraries and cant handle the Ubuntu system dependencies the packages require. Lastly, the global environment wouldn't work if someone isn't on Ubunty, this will lead to conflicts in installing the necessary packages.

- d. A virtual environment seems best suitable for this scenario since the virtual environment will lock our project to Python 3 and its packages. This means that even if Python 4 is installed globally, our current project will still be able to run with Python 3 environment. This allows us to keep projects separate, so we can have on project using Python 3 and another using Python 4. A container does work, but it overkill if the only goal is to preserve a single Python version. A virtual machine is just using too much resources and overkill to spin up a whole other OS just to preserve a Python version. Lastly a global environment wont work since if you decide to upgrade to Python 4, the projects with Python 3 will break.

Problem 3:

Problem 3a.

Created a conda environemnet with Python 3.12 and installed the following packages:

- conda create -n lab1 python=3.12
- conda activate lab1
- conda install neo4j
- conda install python-dotenv
- conda install pandas
- conda install numpy #was already installed
- conda install scipy
- conda install scikit-learn
- conda install requests
- conda install prince
- conda install ipykernel

- conda install conda-forge::wquantiles
- pip install ydata_profiling

In [11]:
```
import numpy as np
import pandas as pd
import weighted # this is a module of wquantiles
from scipy import stats
import prince
from ydata_profiling import ProfileReport
```

Problem 4

The official Python images on Docker Hub use a version of Linux called Debian.However, sometimes you might need to install additional software in a container other than Python and Python packages, and a lot of open source software only works on another version of Linux called Ubuntu.

Part a.

Below is a Dockerfile that builds an image:

In [12]:
```
# Start from the latest Ubuntu image
FROM ubuntu:latest

# Update package lists & install Python 3
RUN apt-get update && \
    apt-get install -y python3 && \
    apt-get clean

#Set the working directory inside the container
WORKDIR /app

#Copy local files into the container
COPY . /app

#By default, launch Python 3
CMD ["python3"]
```

```
  Cell In[12], line 2
    FROM ubuntu:latest
         ^
SyntaxError: invalid syntax
```

Part b.

Proving that the Dockerfile is written correctly by building the image associated with this Dockerfile. Below is the output of the build from my terminal: Note: The image is titled "lab1-image"

- REPOSITORY TAG IMAGE ID CREATED SIZE

- lab1-image latest 05f9921b4ffb 30 seconds ago 299MB
- chandres/jupyterdocker latest d5ec3507b9c9 2 weeks ago 3.36GB
- chandres/python_jupyterdocker latest d5ec3507b9c9 2 weeks ago 3.36GB
- jupyterdocker latest d5ec3507b9c9 2 weeks ago 3.36GB
- mysql latest 94254b456a6d 2 weeks ago 1.27GB
- postgres latest feff5b24fedd 2 weeks ago 662MB
- mongo latest cf340b1e5283 2 weeks ago 1.2GB

I also got the output below when building the image from the terminal (this was the output prior to the one above):

[+] Building 0.5s (9/9) FINISHED docker:desktop-linux => [internal] load build definition from Dockerfile 0.0s => => transferring dockerfile: 374B 0.0s => [internal] load metadata for docker.io/library/ubuntu:latest 0.4s => [internal] load .dockerignore 0.0s => => transferring context: 2B 0.0s => [1/4] FROM docker.io/library/ubuntu:latest@sha256:353675e2a41babd526e2b837d7ec780c2a05bca016 0.0s => => resolve docker.io/library/ubuntu:latest@sha256:353675e2a41babd526e2b837d7ec780c2a05bca016 0.0s => [internal] load build context 0.0s => => transferring context: 14.19kB 0.0s => CACHED [2/4] RUN apt-get update && apt-get install -y python3 && apt-get clean 0.0s => CACHED [3/4] WORKDIR /app 0.0s => [4/4] COPY . /app 0.0s => exporting to image 0.1s => => exporting layers 0.0s => => exporting manifest sha256:be5bd1ec956001af83b30fe83055df0aa5089f34f7db32d6457e129aae0af321 0.0s => => exporting config sha256:d8ab6517f4f61d71d75780b0d0a1da353664fe24b0c19728758d7e56046b7c62 0.0s => => exporting attestation manifest sha256:ad4043f0541fb17b7ccbc796b5cd8b3acbae48b8c26b5c73aca92ce732bc062d 0.0s => => exporting manifest list sha256:8b1c39aa1c8f4b7d65155e9dfc2e5d9deebfbcecac0370d3bbbf4afcd3915fa3 0.0s => => naming to docker.io/library/lab1-image:latest 0.0s => => unpacking to docker.io/library/lab1-image:latest 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/6djciu7txzwci6hbcp62etxmt

Part c.

Below is my conifrmation that the Python Prompt appears correctly:

In [ ]:
```
(base) andrescastellanos@Andress-MacBook-Air DS6600_lab1 % docker run -it la

Python 3.12.3 (main, Aug 14 2025, 17:47:21) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Problem 5

Below is the passage from the docker image running Nethack:

"It is written in the Book of Hermes:

```
After the Creation, the cruel god Moloch rebelled
against the authority of Marduk the Creator.
Moloch stole from Marduk the most powerful of all
the artifacts of the gods, the Amulet of Yendor,
and he hid it in the dark cavities of Gehennom, the
Under World, where he now lurks, and bides his time.
```

Your god Hermes seeks to possess the Amulet, and with it to gain deserved ascendance over the other gods.

You, a newly trained Rhizotomist, have been heralded from birth as the instrument of Hermes. You are destined to recover the Amulet for your deity, or die in the attempt. Your hour of destiny has come. For the sake of your god, you must descend into the depths of Gehennom, face its horrors, and retrieve the Amulet of Yendor. Only then can you ascend to the heavens and join your deity in eternal glory."

Problem 6

We've discussed connecting to MySQL, PostgreSQL, and Mongo using Docker, but there are many different kinds of database systems for different situations, and as a data scientist who can hang with the data engineers, you will need to be able to set up a database from a system you haven't used before. For this problem, use your Docker skills and your navigation of available documentation to create a local Python connection to a graph database running on Neo4j. See this [AWS blog[(https://aws.amazon.com/compare/the-difference-between-graph-and-relational-database/)] or the textbook for a deeper discussion of graph databases.

Part a.

In [ ]:
```
from neo4j import GraphDatabase
import dotenv
import os
```

Part B.

- The Default ports are the following:
    - Bolt Port: 7687

- HTTP Port: 7474

These ports were found in the Docker Hub official Neo4j image page, under the "How to use this image" section.

- Data folder inside the container: /data

These were found in the same docker hub page, in the same section if you read below the code example you will start to read about volumes. This will list /data as the internal storage folder.

- For environmental variables you need to set the NEO4J_AUTH variable to the value of username/password.

  - EX: --env NEO4J_AUTH=neo4j/your_password

If you are working in an enterprise edition you would need the following:

- NEO4J_ACCEPT_LICENSE_AGREEMENT=yes

This was also found in the same docker hub page under the "Getting Started with Docker" page.

Part C. Docker compose file for Neo4j

In [ ]:
```
services:
  neo4j:
    image: neo4j:latest
    container_name: neo4j
    ports:
      - "7474:7474"   # HTTP
      - "7687:7687"   # Bolt
    env_file:
      - .env          # Load environment variables from .env
    volumes:
      - neo4jdata:/data

volumes:
  neo4jdata:
```

```
  Cell In[35], line 1
    services:
             ^
SyntaxError: invalid syntax
```

Part D. Launching the Neo4j container to become a rockstar engineer

In [ ]:
```
from neo4j import GraphDatabase
from dotenv import load_dotenv
import os

# Load environment variables from .env file
```

```
load_dotenv()

# Get Neo4j credentials from environment variable
NEO4J_AUTH = os.getenv('NEO4J_AUTH').split('/')
URI = 'bolt://localhost:7687'
USERNAME = NEO4J_AUTH[0]
PASSWORD = NEO4J_AUTH[1]

# Print credentials for debugging
print(f'Username: {USERNAME}, Password: {PASSWORD}')

try:
    # Create a Driver instance
    driver = GraphDatabase.driver(URI, auth=(USERNAME, PASSWORD))

    # Verify connectivity immediately
    driver.verify_connectivity()
    print('Connection to Neo4j established successfully.')
except Exception as e:
    print(f'Failed to connect to Neo4j: {e}')
finally:
    # Close the driver to release resources
    if 'driver' in locals() and driver:
        driver.close()
```

```
Username: neo4j, Password: BOJACK123
Connection to Neo4j established successfully.
```