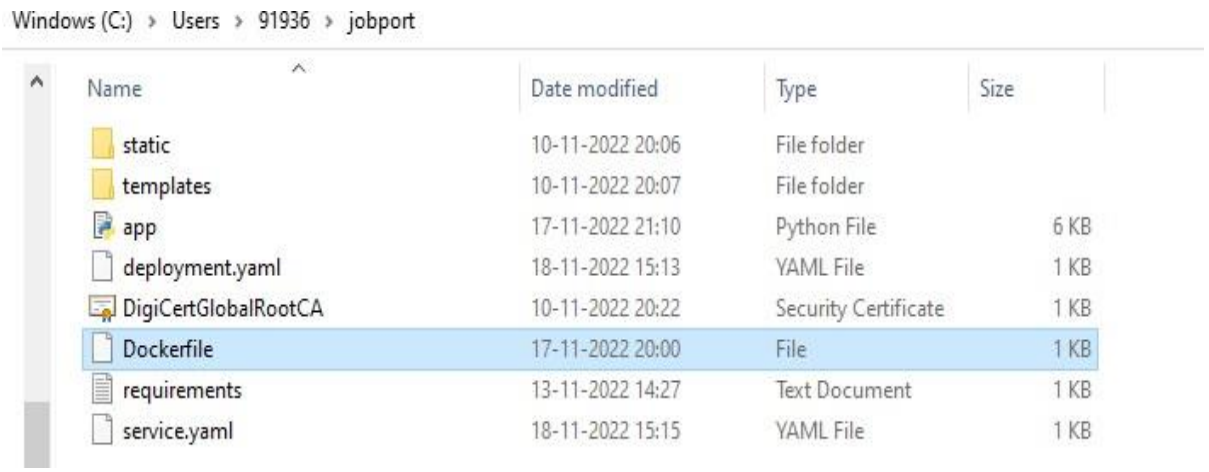
Containerize the App

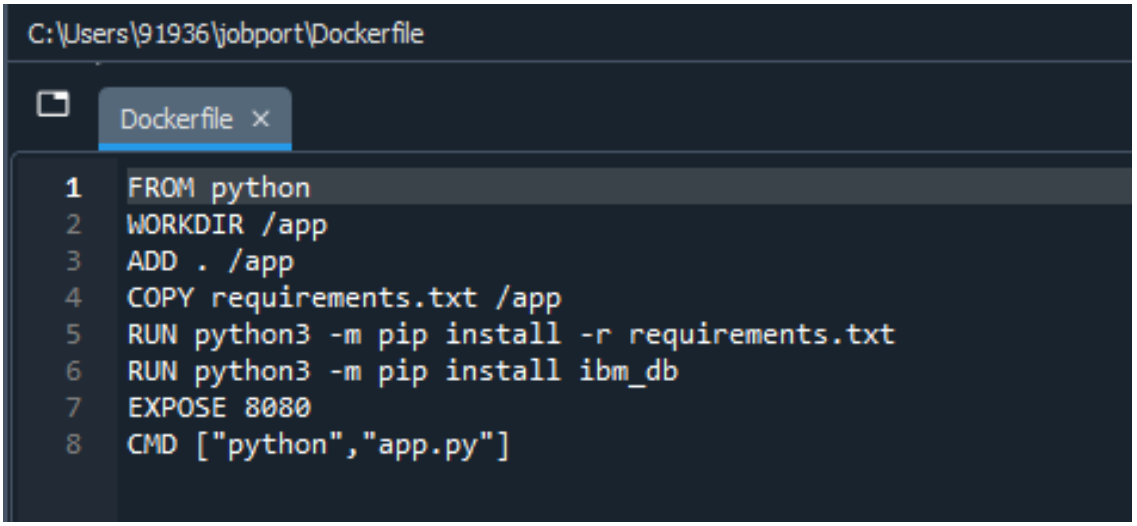
Docker Image Creation for Our Application

Date	15 November 2022
Team ID	**PNT2022TMID23042**
Project Name	**Containment Zone Alerting Application**

Step 1. In our project directory, we created the file named "**Dockerfile**" with no extension-A "Dockerfile" is used to indicate to Docker a base image, the Docker settings you need, and a list of commands you would like to have executed to prepare and start your new container.



Step 2. In the file, the following codes are written



Explanation and breakdown of the above Dockerfile code:

FROM python → Because this Flask application uses Python , we want an environment that supports it and already has it installed.

WORKDIR /app

ADD . /app

COPY requirements.txt/app

➔ Now it's time to add the Flask application to the image. For simplicity, copy the application under the /app directory on our Docker Image.WORKDIR is essentially a **cd** in bash, and COPY copies a certain directory to the provided directory in an image. ADD is another command that does the same thing as COPY, but it also allows you to add a repository from a URL.

RUN python3 -m pip install -r requirements.txt

→ Now that we have our repository copied to the image, we will install all of our dependencies, which is defined in the requirements.txt part of the code.

RUN python3 -m pip install ibm_db

→We used ibm_db as the database so we will install ibm_db

EXPOSE 8080

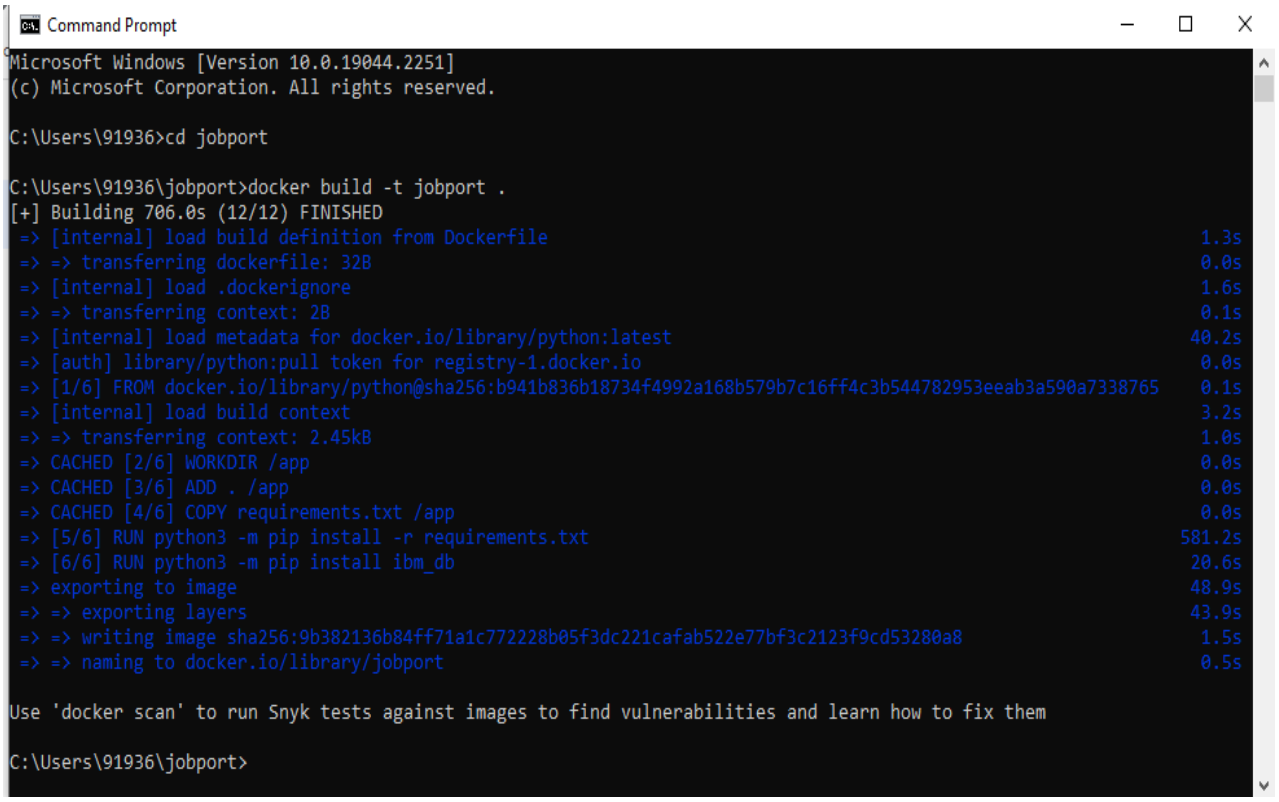
→ We want to expose the port(8080) the Flask application runs on, so we use EXPOSE.

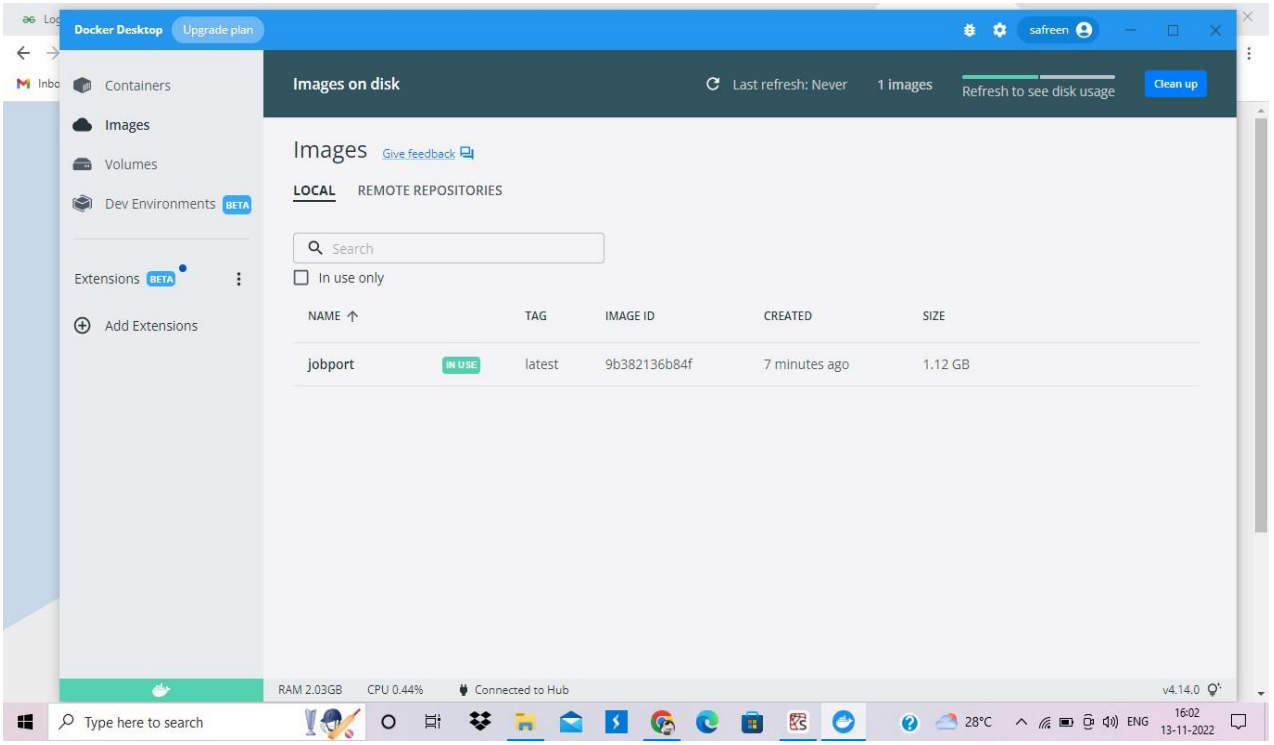
CMD [“python”, “app.py”]

→ ENTRYPOINT specifies the entrypoint of your application.

Step 3: Build an image from the Dockerfile

Open the terminal and type this command to build an image from your Dockerfile:
docker build -t <image_name>:<tag>


Command Prompt
Microsoft Windows [Version 10.0.19044.2251]
(c) Microsoft Corporation. All rights reserved.
C:\Users\91936>cd jobport
C:\Users\91936\jobport>docker build -t jobport .
[+] Building 706.0s (12/12) FINISHED
=> [internal] load build definition from Dockerfile 1.3s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 1.6s
=> => transferring context: 2B 0.1s
=> [internal] load metadata for docker.io/library/python:latest 40.2s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [1/6] FROM docker.io/library/python@sha256:b941b836b18734f4992a168b579b7c16ff4c3b544782953eeab3a590a7338765 0.1s
=> [internal] load build context 3.2s
=> => transferring context: 2.45kB 1.0s
=> CACHED [2/6] WORKDIR /app 0.0s
=> CACHED [3/6] ADD . /app 0.0s
=> CACHED [4/6] COPY requirements.txt /app 0.0s
=> [5/6] RUN python3 -m pip install -r requirements.txt 581.2s
=> [6/6] RUN python3 -m pip install ibm_db 20.6s
=> exporting to image 48.9s
=> => exporting layers 43.9s
=> => writing image sha256:9b382136b84ff71a1c772228b05f3dc221cafab522e77bf3c2123f9cd53280a8 1.5s
=> => naming to docker.io/library/jobport 0.5s
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
C:\Users\91936\jobport>


Docker Desktop
Upgrade plan
safreen
Containers
Images
Volumes
Dev Environments
BETA
Extensions
BETA
Add Extensions
Images on disk
Last refresh: Never
1 images
Refresh to see disk usage
Clean up
Images
Give feedback
LOCAL
REMOTE REPOSITORIES
Search
In use only
NAME
TAG
IMAGE ID
CREATED
SIZE
jobport
IN USE
latest
9b382136b84f
7 minutes ago
1.12 GB
RAM 2.03GB
CPU 0.44%
Connected to Hub
v4.14.0
Type here to search
28°C
ENG
16:02
13-11-2022

Step 4: Run your container locally and test

After you build your image succesfully, type: docker run -d -p 8080:8080 jobport

This command will create a container that contains all the application code and dependencies from the image and runs it locally.