Lab Assignment 2: Dockerfile & Containerizing Applications

Name: Aarohan Sarkar
Reg. No.: 12213072

Objective

- Learn how to create and use a Dockerfile.
- Containerize a simple Flask application using Docker.
- Practice building custom images and running them.
- Implement data persistence and access container logs.

---

1. Dockerfile Creation & Explanation (10 Marks)

Application: A simple Flask web server that runs on port 80.

```
from flask import Flask
import logging

app = Flask(__name__)

logging.basicConfig(filename='/logs/output.log', level=logging.INFO)
app.logger.info("Flask app started successfully")

@app.route("/")
def home():
    return "Hello from Flask with logging!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80)
```

Dockerfile:

```
# Use a lightweight Python base image
FROM python:3.9-slim

# Set the working directory in the container
WORKDIR /app

# Copy your code and requirements to the container
COPY . /app

# Install dependencies
RUN pip install -r requirements.txt

# Create logs directory inside the container
RUN mkdir -p /logs

# Open port 80 so the app can be accessed
EXPOSE 80

# Start the Flask app when the container starts
CMD ["python", "app.py"]
```

Explanation of Dockerfile Instructions:

- FROM python:3.9-slim: Uses a minimal Python image as the base.
- WORKDIR /app: Sets the working directory inside the container.
- COPY . /app: Copies all application files into the container.
- RUN pip install -r requirements.txt: Installs Python dependencies.
- RUN mkdir -p /logs: Creates a directory for application logs.
- EXPOSE 80: Tells Docker to open port 80.
- CMD ["python", "app.py"]: Runs the Flask app on container startup.

---

2. Building and Running Custom Images (10 Marks)

Image Build:

docker build -t flask-docker-app .

Container Run:

```
C:\Users\ASUS\Desktop\lab2-docker>docker run -d -p 8080:80 flask-docker-app
e70392711f48912658d766b045d674487dc751a96e11fc8b571dc1bbea0a64c6

C:\Users\ASUS\Desktop\lab2-docker>docker ps
CONTAINER ID   IMAGE              COMMAND           CREATED         STATUS         PORTS                  NAMES
e70392711f48   flask-docker-app   "python app.py"   5 seconds ago   Up 4 seconds   0.0.0.0:8080->80/tcp   upbeat_haibt
```

docker run -d -p 8080:80 flask-docker-app

Verification:

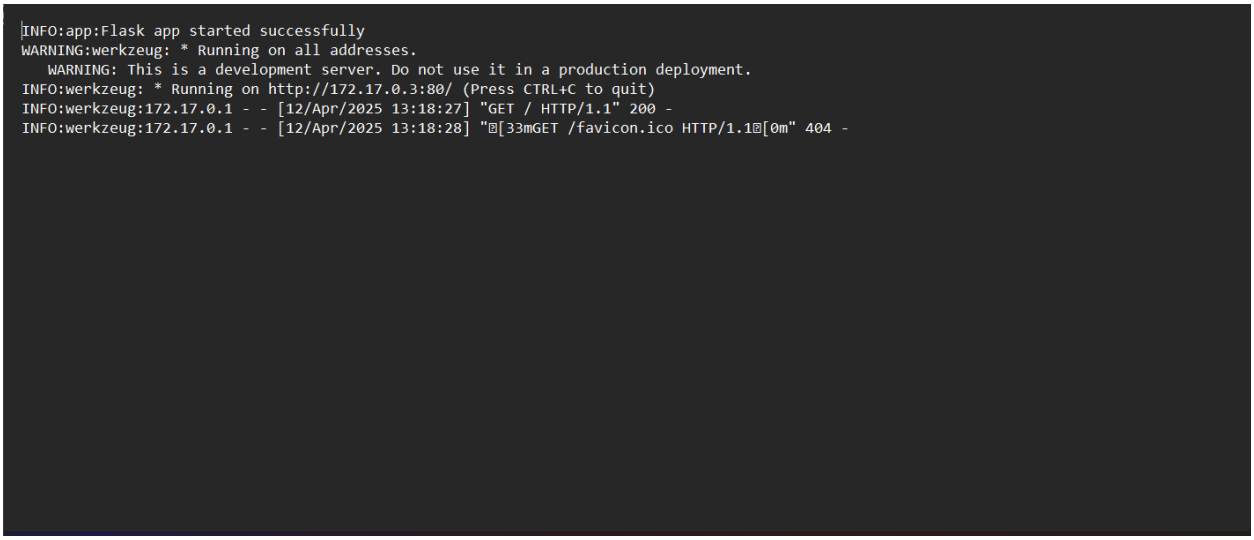
- Successfully opened http://localhost:8080 in a web browser.
- Screenshot captured for evidence.

---

3. Persistence & Logs (10 Marks)

Volume Mount:

docker run -d -p 8080:80 -v ${PWD}/logs:/logs flask-docker-app



Inspect Logs:

cat logs/output.log

Log Purpose:

- Logs are written to /logs/output.log by the application using Python's logging module.
- This helps in debugging runtime issues and confirming expected behavior.

Screenshots Included:

- Terminal showing docker build command and successful image creation.
- Terminal showing docker ps with running container.

- Web browser showing the Flask app running at http://localhost:8080.
- Terminal showing log output or contents of the logs folder.

Conclusion

This lab successfully demonstrated the use of Docker to containerize a Python Flask application, implement data persistence using volumes, and troubleshoot via logs. The app was built, deployed, and tested successfully.