DevOps Lab Program-1 Documentation

Title:

Build a Docker Container from a Custom Dockerfile

Objective:

To create a custom Docker image using a Flask application and run it inside a container.

Software Requirements:

- Docker
- Python 3
- Flask (specified in requirements.txt)

Files Used:

1. **app.py** – A simple Flask web application.

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Docker container running successfully!"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=9090)
```

Fig-01: app.py file containing the statement to be displayed in the browser

2. **requirements.txt**

```
flask
```

Fig-02: requirements.txt file containing the necessary installation software

3. **Dockerfile**

```
#Base image
FROM python:3.9-slim

#Working directory
WORKDIR /app

#Copy files from local system to container
COPY . /app

#Install the dependencies
RUN pip install --no-cache-dir -r requirements.txt || true

#Run the container on port 9090
EXPOSE 9090

#Start the application
CMD ["python","app.py"]
```

Fig-03: Dockerfile consisting of base image, commands to install and run the dependencies, setting the port and starting the application

Procedure:

1. Create Project Folder:
Create a directory named Program-1 and place the following files inside it:

app.py, requirements.txt, and Dockerfile.

2. Build Docker Image:
Open terminal inside the folder and run:

```
docker build -t flask-app .
```

3. Verify image creation:

```
docker images
```

4. Run container:

```
docker run -d -p 9090:9090 --name flask-container flask-app
```

5. Check running containers:

```
docker ps
```

6. Access Application:

Open a web browser and go to:

http://localhost:9090

You should see the message:

“Docker container running successfully!”

Result:

Successfully created a custom Docker image from a Flask application and deployed it inside a running container accessible through a web browser.

Conclusion:

This experiment demonstrates how to containerize a simple Python Flask application using Docker, build a custom image, and run it as a portable, isolated containerized service.