DevOps Lab Program 3: Containerization with Docker

PART A: Deploying a WAR File to Tomcat using Docker

Objective:

To create a Maven web project, generate a WAR file, and deploy it using Apache Tomcat inside a Docker container.

Steps:

1. Create a Maven Web App in Eclipse
   - Use Archetype: maven-archetype-webapp
   - Validate index.html and web.xml exist

2. Build the Project
   - Run: Maven clean Maven install
   - Output: target/Myapp3.war

3. Dockerfile:

```
FROM tomcat:9.0

RUN rm -rf /usr/local/tomcat/webapps/*

COPY target/Myapp3.war /usr/local/tomcat/webapps/ROOT.war

EXPOSE 8080
```

4. Build Docker Image:

```
docker build -t myapp3 .
```

5. Run the Container:

```
docker run -d -p 8081:8080 myapp3
```

6. Access at: http://localhost:8081

7. Cleanup:

```
docker ps

docker stop <container_id>

docker rm <container_id>
```

PART B: Multi-Container App using Docker Compose

Objective:

Run two Python services with Docker Compose:

- app1: Flask-based API
- app2: Requests-based client

Folder Structure:

```
multi-container-app/
 app1/
    app.py
    requirements.txt
    Dockerfile
 app2/
    app.py
    requirements.txt
    Dockerfile
```

docker-compose.yml

Code for app1 (Flask):

app1/app.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello():

    return "Hello from App 1!"

if __name__ == '__main__':

    app.run(host='0.0.0.0', port=5000)
```

app1/requirements.txt:

```
flask==3.0.0
```

app1/Dockerfile:

```
FROM python:3.12-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

EXPOSE 5000

CMD ["python", "app.py"]
```

Code for app2 (Requests):

app2/app.py

```
import requests
```

```
response = requests.get("http://app1:5000/")

print("Response from App 1:", response.text)
```

app2/requirements.txt:

```
requests==2.31.0
```

app2/Dockerfile:

```
FROM python:3.12-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

CMD ["python", "app.py"]
```

docker-compose.yml:

```
version: '3.9'

services:

  app1:

    build: ./app1

    networks:

      - app-network

    ports:

      - "5000:5000"

  app2:

    build: ./app2

    networks:

      - app-network
    depends_on:
      - app1
networks:
  app-network:
    driver: bridge
```

Run the App:

```
docker-compose build
docker-compose up
```

Cleanup:

```
docker-compose down
```

Docker Commands Summary:

```
docker build -t myapp3 .
docker run -d -p 8081:8080 myapp3
docker ps
docker stop <id>
docker-compose build
docker-compose up
docker-compose down
```