Ah got it, Sarvesh! Here's the **full and final clean summary** of **DevOps Program 3: Containerization with Docker** — including both **Part A and Part B**, with all the steps, commands, and code examples. This is ready to be used for lab records or reports.

✅ 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
- Add content to index.html
- Validate web.xml exists

2. Build the Project

Run:

Right-click → Run As → Maven clean

Right-click → Run As → Maven install

✅ WAR file gets created at:
target/<projectname>.war (e.g., target/Myapp3.war)

---

3. Create a 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 the Docker Image

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

5. Run the Docker Container

docker run -d -p 8081:8080 myapp3

Access at: http://localhost:8081

🧹 Optional Cleanup

```
docker ps                  # List containers
docker stop <container_id>  # Stop a container
docker rm <container_id>    # Remove a container
```

🔹 PART B: Multi-Container App using Docker Compose

🎯 Objective:

Use Docker Compose to run two Python-based services:

- app1: Flask-based service provider
- app2: Requests-based service consumer

✅ Folder Structure

```
multi-container-app/
 ├── app1/
 |   ├── app.py
 |   ├── requirements.txt
 |   └── Dockerfile
 ├── app2/
 |   ├── app.py
 |   ├── requirements.txt
 |   └── Dockerfile
 └── docker-compose.yml
```

✅ Code for app1 (Flask server)

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 client)**

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
```

You’ll see:

```
Response from App 1: Hello from App 1!
```

And visiting http://localhost:5000 in your browser shows:
Hello from App 1!

🧹 Cleanup

docker-compose down

🔁 Summary of Key Docker Commands

Command	Purpose
docker build -t myapp3 .	Build image
docker run -d -p 8081:8080 myapp3	Run container
docker ps	Show running containers
docker stop <id>	Stop a container
docker-compose build	Build multi-container setup
docker-compose up	Run all containers
docker-compose down	Stop all containers and remove networks

Let me know if you want this in **PDF or DOCX** format for submission, or help with Docker Hub/GitHub integration! 💪 🐳