Java WAR Deployment on Tomcat using Docker

Student Handout – Deployment Strategies and Best Practices

1. Overview

This handout explains three different approaches to deploy a Java WAR file into a Tomcat container using Docker, including advantages, drawbacks, and best practices.

2. Deployment Option 1 – Build WAR Outside Docker

Flow:

1. 1. Build WAR using Maven on local machine or VM
2. 2. Create Docker image using Tomcat base image
3. 3. Copy WAR into Tomcat webapps directory
4. 4. Run Tomcat container

Dockerfile Example:

```
FROM tomcat:9.0-jdk17-temurin
COPY target/myapp.war /usr/local/tomcat/webapps/myapp.war
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

Commands:

```
mvn clean package -DskipTests
docker build -t myapp:opt1 .
docker run -p 8080:8080 myapp:opt1
```

3. Deployment Option 2 – Build WAR Inside Docker (Single Stage)

Problem: Maven dependencies in .m2 increase image size significantly.

```
FROM maven:3.9-eclipse-temurin-17
WORKDIR /app
COPY . .
RUN mvn clean package -DskipTests
```

4. Deployment Option 3 – Multi-Stage Build (Recommended)

Stage 1 builds the WAR. Stage 2 runs Tomcat with only the WAR file.

```
# ---------- Stage 1: Build ----------
FROM maven:3.9-eclipse-temurin-17 AS builder
WORKDIR /app
COPY pom.xml .
RUN mvn -q -DskipTests dependency:go-offline
COPY src ./src
RUN mvn -q clean package -DskipTests

# ---------- Stage 2: Runtime ----------
FROM tomcat:9.0-jdk17-temurin
RUN rm -rf /usr/local/tomcat/webapps/*
COPY --from=builder /app/target/*.war
/usr/local/tomcat/webapps/ROOT.war
EXPOSE 8080
CMD ["catalina.sh", "run"]
```

Build and Run:

```
docker build -t myapp:opt3 .
docker run -p 8080:8080 myapp:opt3
```

5. Comparison Summary

Option	Build Location	Image Size	Best For
Option 1	Outside Docker	Small	When CI already builds WAR
Option 2	Inside Docker (Single)	Large	Learning/Demo only
Option 3	Inside Docker (Multi-Stage)	Small	Production and CI/CD