
🧪 Lab Guide: Jenkins CI/CD Pipeline for Java App Deployment on Tomcat

🎯 Objective

Set up a Jenkins pipeline that:

1. Clones a Java project from GitHub
2. Builds it using Maven
3. Scans code quality using SonarQube
4. Deploys the built artifact (`.war`) to Apache Tomcat

🧩 Prerequisites

Component	Description
OS	Ubuntu 22.04 (or similar Linux)
Jenkins	Installed and running on port 8080
Maven	Installed and configured in Jenkins
Java (JDK 11+)	Installed on Jenkins server
Git	Installed
SonarQube	Running locally or remotely (with access token)

Tomcat	Installed and running on port 8081 or 8080

🧰 Step 1: Install and Configure Jenkins

```
# Update system
sudo apt update -y

# Install Java
sudo apt install openjdk-17-jdk -y

# Add Jenkins repository key and install Jenkins
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update -y
sudo apt install jenkins -y
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

Then open Jenkins in your browser:
➡️ **http://:8080**

Retrieve the initial admin password:

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Install suggested plugins and create an admin user.

⚙️ Step 2: Configure Jenkins Tools

In Jenkins Dashboard → **Manage Jenkins** → **Tools Configuration**

Tool	Configuration
JDK	Name: jdk17 → /usr/lib/jvm/java-17-openjdk-amd64
Maven	Name: maven3 → Install automatically
Git	Use system Git
SonarQube Scanner	Install automatically

🧩 Step 3: Integrate SonarQube

1. Go to **Manage Jenkins** → **System** → **SonarQube Servers**
2. Add:
   - Name: SonarQube
   - Server URL: http://<sonarqube-server>:9000
   - Authentication Token: (generate from SonarQube UI → My Account → Security)
3. Add Sonar Scanner:
   - **Manage Jenkins** → **Tools** → **SonarQube Scanner**

    - Name: sonar-scanner
    - Install automatically

Verify with:

```
sonar-scanner -v
```

4.

📦 Step 4: Install Deploy to Container Plugin

In Jenkins:

- Go to **Manage Jenkins → Plugins → Available**
- Search and install **Deploy to container** plugin
- Restart Jenkins

🏗️ Step 5: Create a New Jenkins Job

1. **New Item → Maven Project**
2. Name: JavaApp-CI-CD
3. Click **OK**

General Configuration

✅ Check: “Discard Old Builds”
✅ GitHub Project (optional)

Source Code Management

- Git → Repository URL:
  https://github.com/<username>/<repo>.git

- Branch: */main

⚒️ Step 6: Build Configuration

Add Build Steps

🧱 Step 1: Clean and Build with Maven

Goals: clean package
POM: pom.xml

🧠 Step 2: SonarQube Analysis

Add a **Build Step** → **Execute SonarQube Scanner**

Example properties:

```
sonar.projectKey=my-java-app
sonar.projectName=My Java App
sonar.sources=src
sonar.java.binaries=target
sonar.host.url=http://<sonarqube-server>:9000
sonar.login=<your-token>
```

🚀 Step 3: Deploy WAR to Tomcat

Add **Post-build Action** → **Deploy war/ear to a container**

- WAR/EAR files: `**/target/*.war`
- Context path: `/myapp`
- Containers:
  - Tomcat 9.x Remote
  - Credentials → Add (username/password of Tomcat user)
  - URL: `http://<tomcat-server>:8080`

🪄 Step 7: Configure Tomcat for Remote Deployment

On Tomcat server, edit:

```
sudo nano /opt/tomcat/conf/tomcat-users.xml
```

Add this inside `<tomcat-users>`:

```
<role rolename="manager-script"/>
<user username="admin" password="admin123" roles="manager-script"/>
```

Then restart Tomcat:

```
sudo systemctl restart tomcat
```

Verify deployment access:
👉 http://:8080/manager/html

▶️ Step 8: Run the Pipeline

Click **Build Now** in Jenkins.

Monitor the console output — you should see:

```
Cloning repository...
Building project with Maven...
SonarQube analysis completed...
Deploying to Tomcat...
BUILD SUCCESS
```

Visit:
➡️ **http://:8080/myapp**

You'll see your deployed application 🎉

✅ Step 9: Verify SonarQube Dashboard

Open SonarQube UI → Projects → *My Java App*
You'll see:

- Code coverage
- Bugs/Vulnerabilities
- Code smells

- Security issues

🔐 Optional Enhancements

Enhancement	Description
Email Notifications	Add post-build action → "Email Notification"
Quality Gates	Fail build if SonarQube quality gate fails
Parameterized Builds	Allow dynamic branch selection
Blue Ocean View	Install “Blue Ocean” plugin for visual pipelines

---

📘 Summary

✅ Jenkins automates:

1. **Code Fetching** from GitHub
2. **Building** via Maven
3. **Testing/Analysis** via SonarQube
4. **Deployment** to Tomcat

This forms a **complete standalone CI/CD workflow** 🧠

Would you like me to:

- 📄 Format this into a **downloadable lab PDF** (with screenshots placeholders), or
- 🧰 Convert it into a **Jenkinsfile-based pipeline (Declarative Pipeline)** version?

That way, you can choose between GUI-based setup or code-driven automation.