Perfect 👍 — I'll give you a **clean, final DevOps reference** you can reuse anytime.

This includes:

✅ Full Declarative Pipeline

✅ ALL fixes we applied

✅ Correct Maven Sonar usage

✅ Correct Nexus config

✅ Correct Tomcat deploy

✅ Slack notification

✅ Verification commands (so you never troubleshoot blindly again)

You can literally bookmark this.

## 🚀 FINAL FULL PIPELINE (COPY-PASTE SAFE)

```
pipeline {
 agent any

 stages {

  stage('Git Clone') {
   steps {
    git branch: 'feature-1.1',
    url: 'https://github.com/betawins/sabear_simplecutomerapp.git'
   }
  }

  stage('SonarQube Integration') {
```

```
  steps {
    withSonarQubeEnv('SonarQube') {
      sh 'mvn sonar:sonar'
    }
  }
}

stage('Maven Compilation') {
  steps {
    sh 'mvn clean install -DskipTests'
  }
}

stage('Nexus Artifactory') {
  steps {
    nexusArtifactUploader(
      nexusVersion: 'nexus3',
      protocol: 'http',
      nexusUrl: 'YOUR_NEXUS_PRIVATE_IP:8081',
      groupId: 'com.javatpoint',
      version: "${env.BUILD_NUMBER}-SNAPSHOT",
      repository: 'Project-02',
      credentialsId: 'Nexus',
      artifacts: [
        [artifactId: 'SimpleCustomerApp',
         classifier: '',
         file: "target/SimpleCustomerApp-${env.BUILD_NUMBER}-SNAPSHOT.war",
         type: 'war']
      ]
      )
     }
    }

    stage('Deploy On Tomcat') {
     steps {
      sh '''
      docker cp target/SimpleCustomerApp-${BUILD_NUMBER}-SNAPSHOT.war tomcat-
container:/usr/local/tomcat/webapps/
      '''
     }
    }

  }

  post {
   success {
     slackSend channel: '#jenkins', message: "Build SUCCESS ${env.JOB_NAME} #${env.BUILD_NUMBER}"
   }
   failure {
     slackSend channel: '#jenkins', message: "Build FAILED ${env.JOB_NAME} #${env.BUILD_NUMBER}"
   }
  }
}
```

# 🧠 WHAT EACH STAGE DOES (SUPER SIMPLE)

## ✅ Git Clone

Downloads code into:

```
/var/jenkins_home/workspace/<job>
```

## ✅ SonarQube Integration (IMPORTANT FIX)

We use:

```
mvn sonar:sonar
```

NOT sonar-scanner.

Why?

```
Maven project → Maven provides build context → better analysis.
```

Flow:

```
Maven prepares project
→ sends data
→ SonarQube server checks quality
```

## ✅ Maven Compilation

```
mvn clean install
```

Creates WAR:

```
target/SimpleCustomerApp-<BUILD_NUMBER>-SNAPSHOT.war
```

## ☑ Nexus Artifactory (FIXES APPLIED)

Important things we fixed:

✓ Correct credential ID:

```
credentialsId: 'Nexus'
```

✓ Correct Nexus URL:

```
PRIVATE_IP:8081
```

✓ Correct artifact name from pom.xml.

## ☑ Deploy On Tomcat (CRITICAL FIX)

❌ OLD (caused issues)

```
docker cp target/*.war ...
```

✓ NEW (safe)

```
docker cp target/SimpleCustomerApp-${BUILD_NUMBER}-SNAPSHOT.war ...
```

Avoids deploying multiple WARs.

## ☑ Slack Notification

Runs automatically:

```
success → Slack message
failure → Slack alert
```

## 🔎 PROVING / VERIFICATION CHECKLIST (SAVE THIS)

Whenever pipeline runs, verify in THIS order.

### 1️⃣ Check WAR inside Jenkins container

```
docker exec -it jenkins-container ls /var/jenkins_home/workspace/<job>/target/
```

Expected:

```
SimpleCustomerApp-XX-SNAPSHOT.war
```

### 2️⃣ Check WAR inside Tomcat

```
docker exec -it tomcat-container ls /usr/local/tomcat/webapps/
```

Expected:

```
SimpleCustomerApp-XX-SNAPSHOT
SimpleCustomerApp-XX-SNAPSHOT.war
```

Folder presence = deployment success.

### 3️⃣ Check Tomcat logs if needed

```
docker logs tomcat-container | tail -50
```

## 4️⃣ Correct URL Format (VERY IMPORTANT)

Tomcat uses WAR filename as context path.

So open:

```
http://TOMCAT_IP:8082/SimpleCustomerApp-XX-SNAPSHOT
```

NOT:

```
/SimpleCustomerApp
```

## 5️⃣ If You See 404

It means:

```
App has no index.jsp or root servlet.
```

NOT pipeline failure.

## 6️⃣ Clean Tomcat (Only if things look weird)

```
docker exec -it tomcat-container rm -rf /usr/local/tomcat/webapps/*
```

Then rebuild.

## 🎯 FULL FLOW (YOUR FINAL ARCHITECTURE)

```
GitHub
   ↓
Jenkins Container
   ↓
mvn sonar:sonar → SonarQube EC2
```

```
   ↓
mvn clean install → WAR
   ↓
Upload → Nexus EC2
   ↓
docker cp → Tomcat Container
   ↓
Slack Notification
```

## ⭐ BIGGEST LESSONS / FIXES WE APPLIED

✓ Use `mvn sonar:sonar` for Maven projects

✓ Tool names must match EXACTLY

✓ SonarQube server name must match EXACTLY

✓ Use PRIVATE Nexus IP

✓ Use credential ID (not username)

✓ Deploy single WAR, not wildcard

✓ Tomcat URL = WAR filename

👉 **Interview one-liner:**

“This declarative Jenkins pipeline performs end-to-end CI/CD by cloning code, running Maven-based SonarQube analysis, building a versioned WAR, uploading to Nexus, deploying to a Dockerized Tomcat, and sending Slack notifications.”