Got you 👍

Here are **both pipelines** clean and ready — plus a **simple explanation** you can use when someone asks what went wrong.

☑ Pipeline 1 — Scripted Pipeline

```
node {

    try {

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

        stage('SonarQube Integration') {
            withSonarQubeEnv('SonarQube') {
                sh 'mvn sonar:sonar'
            }
        }

        stage('Maven Compilation') {
            sh '''
            mkdir -p src/main/webapp
            cp -r WebContent/* src/main/webapp/ || true
            mvn clean install -DskipTests
            '''
        }

        stage('Nexus Artifactory') {
            nexusArtifactUploader(
                nexusVersion: 'nexus3',
                protocol: 'http',
                nexusUrl: '52.207.98.83:8081',
                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') {
            sh """
            docker cp target/SimpleCustomerApp-${env.BUILD_NUMBER}-SNAPSHOT.war \
            tomcat-container:/usr/local/tomcat/webapps/
            """
        }

        slackSend channel: '#jenkins-integration',
        message: "Build SUCCESS ${env.JOB_NAME} #${env.BUILD_NUMBER}"

    } catch (err) {

        slackSend channel: '#jenkins-integration',
        message: "Build FAILED ${env.JOB_NAME} #${env.BUILD_NUMBER}"

        throw err
    }
}
```

☑ Pipeline 2 — Declarative Pipeline

```
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 '''
    mkdir -p src/main/webapp
    cp -r WebContent/* src/main/webapp/ || true
    mvn clean install -DskipTests
    '''
  }
}

stage('Nexus Artifactory') {
  steps {
    nexusArtifactUploader(
      nexusVersion: 'nexus3',
      protocol: 'http',
      nexusUrl: '52.207.98.83:8081',
      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-integration',
      message: "Build SUCCESS ${env.JOB_NAME} #${env.BUILD_NUMBER}"
    }
    failure {
      slackSend channel: '#jenkins-integration',
      message: "Build FAILED ${env.JOB_NAME} #${env.BUILD_NUMBER}"
    }
  }
}
```

🧠 SIMPLE EXPLANATION (How you explain the issue)

👉 What was wrong:

The project is a **legacy Java web app**.

It stores JSP files here:

`WebContent/`

But Maven expects:

`src/main/webapp/`

So Maven kept building an **empty WAR**:

```
WAR size = 1.4 KB
→ No JSP files
→ Tomcat shows 404
```

Nothing was wrong with:

- Jenkins
- Docker
- Tomcat
- Nexus
- SonarQube

The problem was **project structure vs Maven expectations**.

👉 What we fixed:

Inside the pipeline we added:

```
cp WebContent → src/main/webapp
```

So Maven finally packaged:

```
JSP + HTML + CSS
```

WAR size became ~936 KB → app worked.

🎯 SUPER SHORT INTERVIEW ONE-LINER

“I was getting repeated Tomcat 404 errors because the legacy project stored JSPs in WebContent, which Maven ignores. I fixed it by mapping WebContent to src/main/webapp during the Jenkins build so the WAR packaged correctly.”

If you want, I can also give you a 🔥 **10/10 interview-level explanation** (very short but impressive wording) you can literally memorize.