**Challenge-4**

1. **Create one Declarative pipeline job**

Jenkins EC2 Configuration

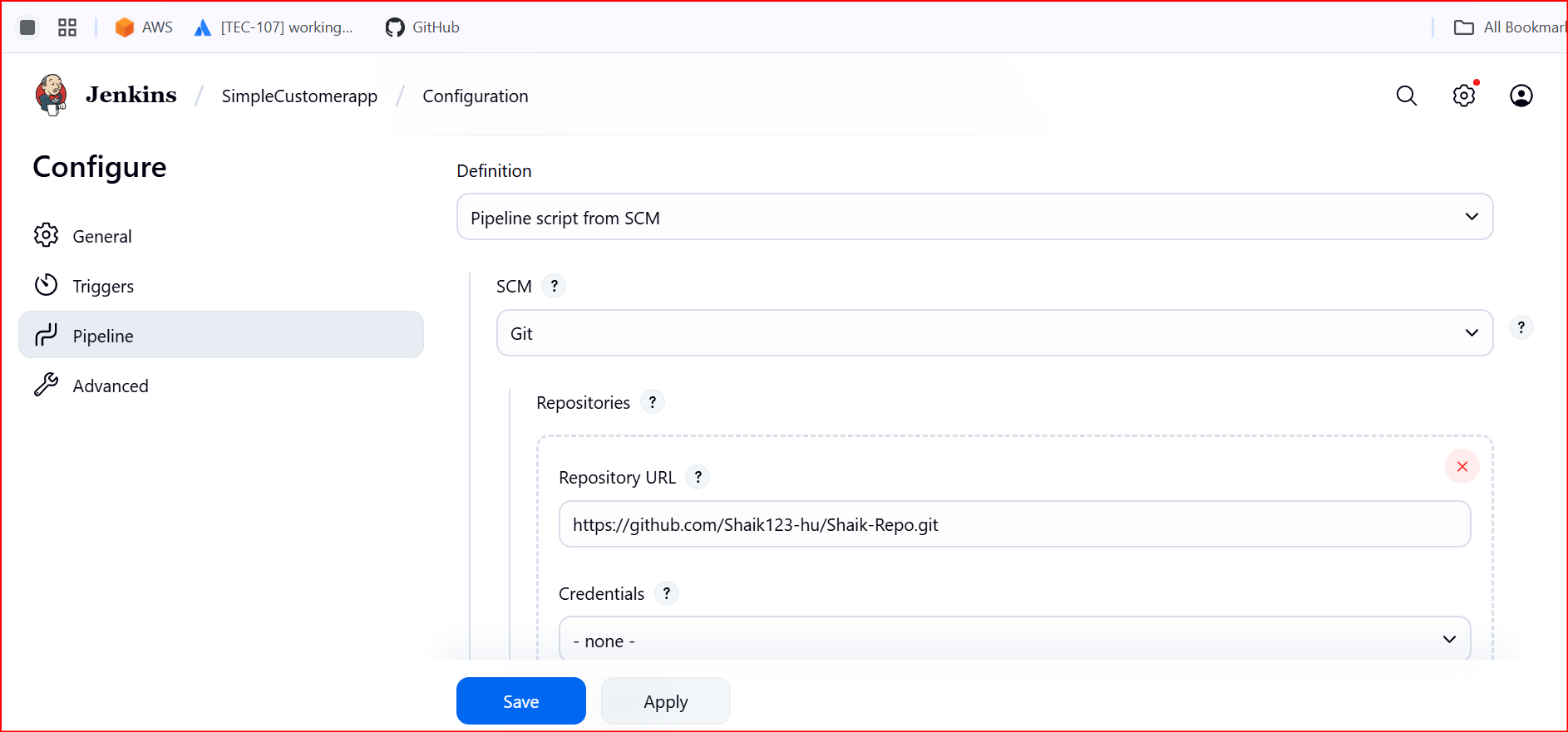
EC2 Instance: Your Jenkins server

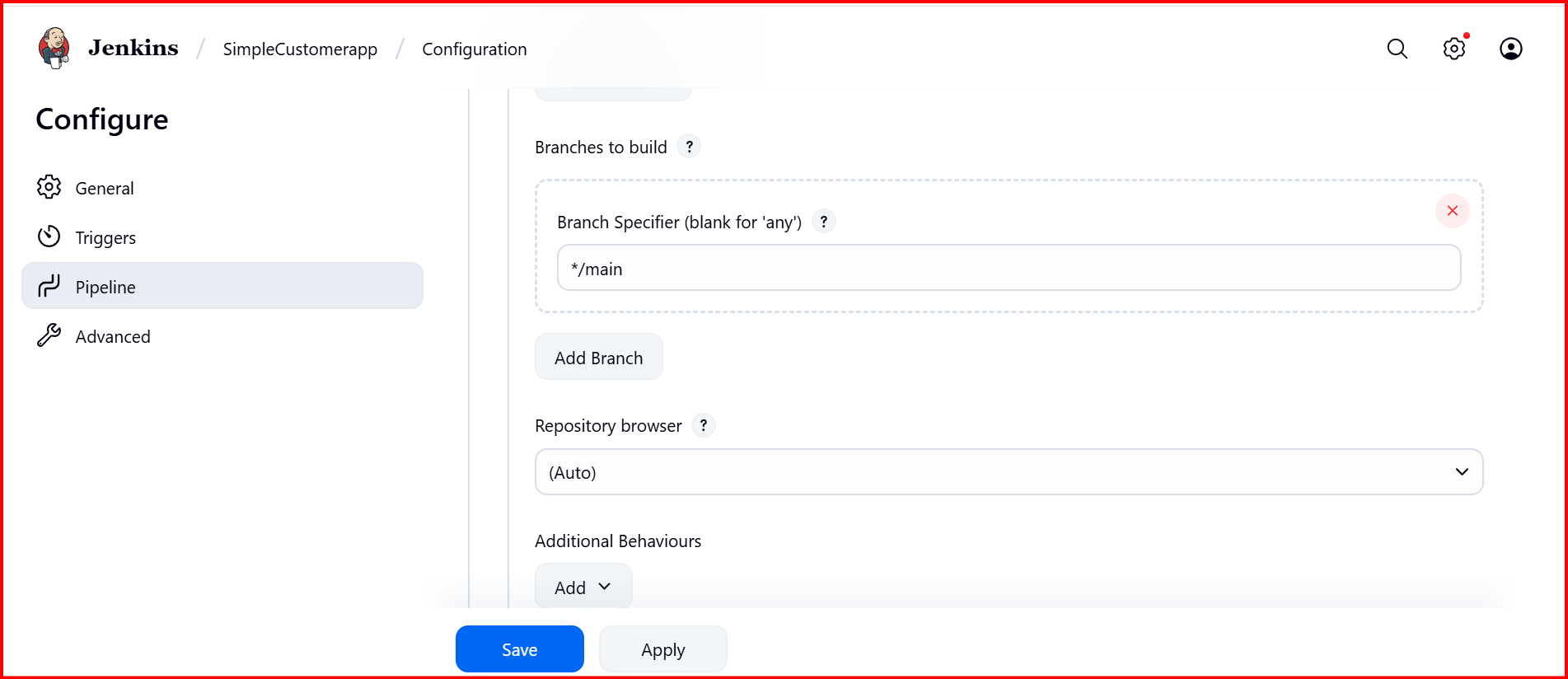
Actions & Configurations:

1. Installed Jenkins  
   * Installed on EC2 (probably Amazon Linux or Ubuntu).
   * Installed Java (required for Jenkins & Maven).
2. Installed Tools in Jenkins  
   * Maven (version 3.8.4) → configured in Manage Jenkins → Global Tool Configuration.
   * JDK (Java 17) → configured for builds.
3. Installed Plugins  
   * Git Plugin → to clone repositories from GitHub.
   * Pipeline Plugin → to create declarative pipelines.
   * Maven Integration Plugin → to run Maven builds.
   * SonarQube Scanner Plugin → to run SonarQube analysis from Jenkins.
   * Slack Notification Plugin → to send Slack messages from Jenkins.
   * Credentials Plugin → to store passwords securely.
   * Nexus artifact loader
   * Stage view plugin

Create the Pipeline Job

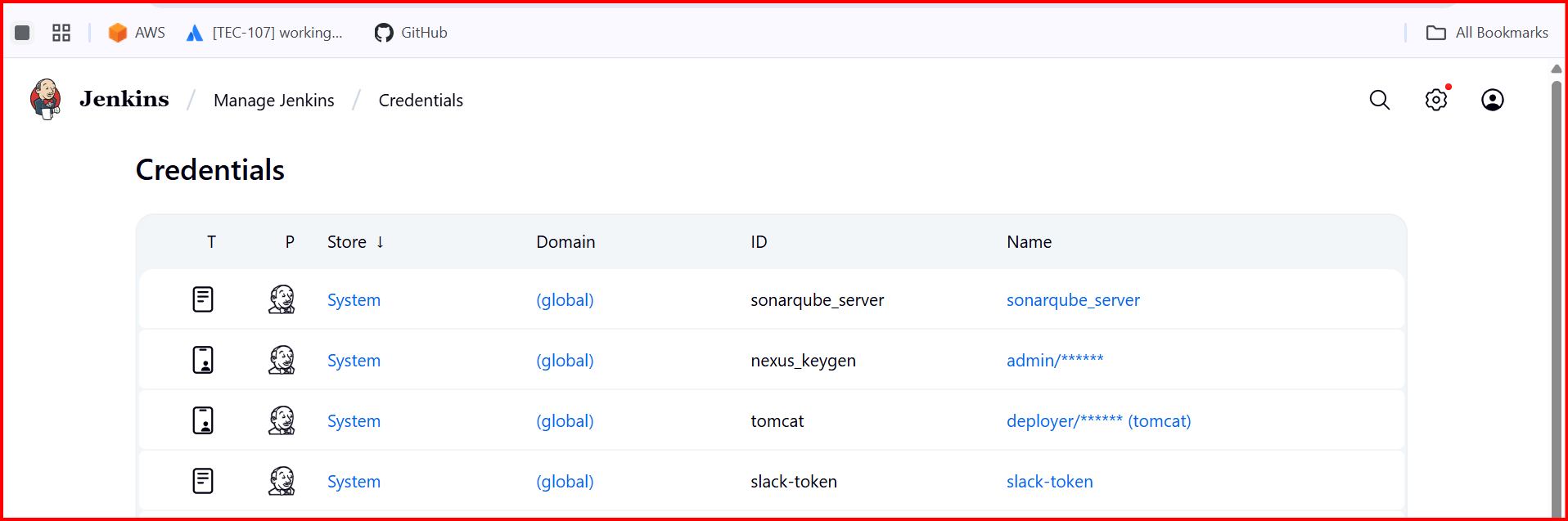
1. *Jenkins → New Item →* Pipeline → name it (e.g. single-declarative-pipeline) → OK.
2. This project is → add: **Declarative pipeline**
   * Choice: BRANCH → main\ndevelop
   * Boolean: RUN\_SONAR → default checked
   * Choice: ENV → dev\nqa\nprod
   * Boolean: DEPLOY → default checked
3. Pipeline section → pick Pipeline script (paste Jenkinsfile below) *or* Pipeline script from SCM if you’ll commit it to your repo.





**Created Credentials:**

**Create Credintials for Sonarqube nexus tomcat**



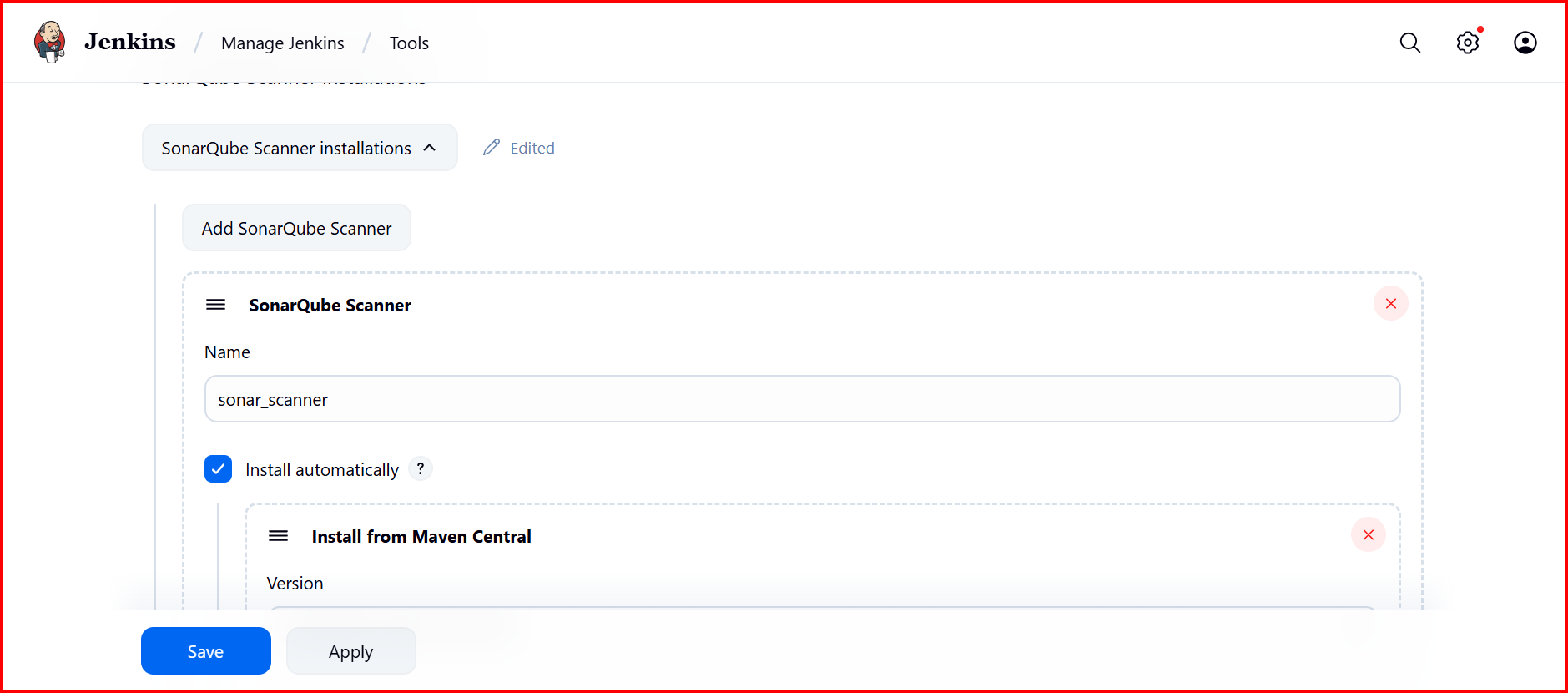
**Sonarqube configuration**:

**1: Install the SonarQube Scanner Plugin**

1. Go to **Manage Jenkins → Plugins → Available**.
2. Search for **SonarQube Scanner**.
3. Install it and restart Jenkins if required.

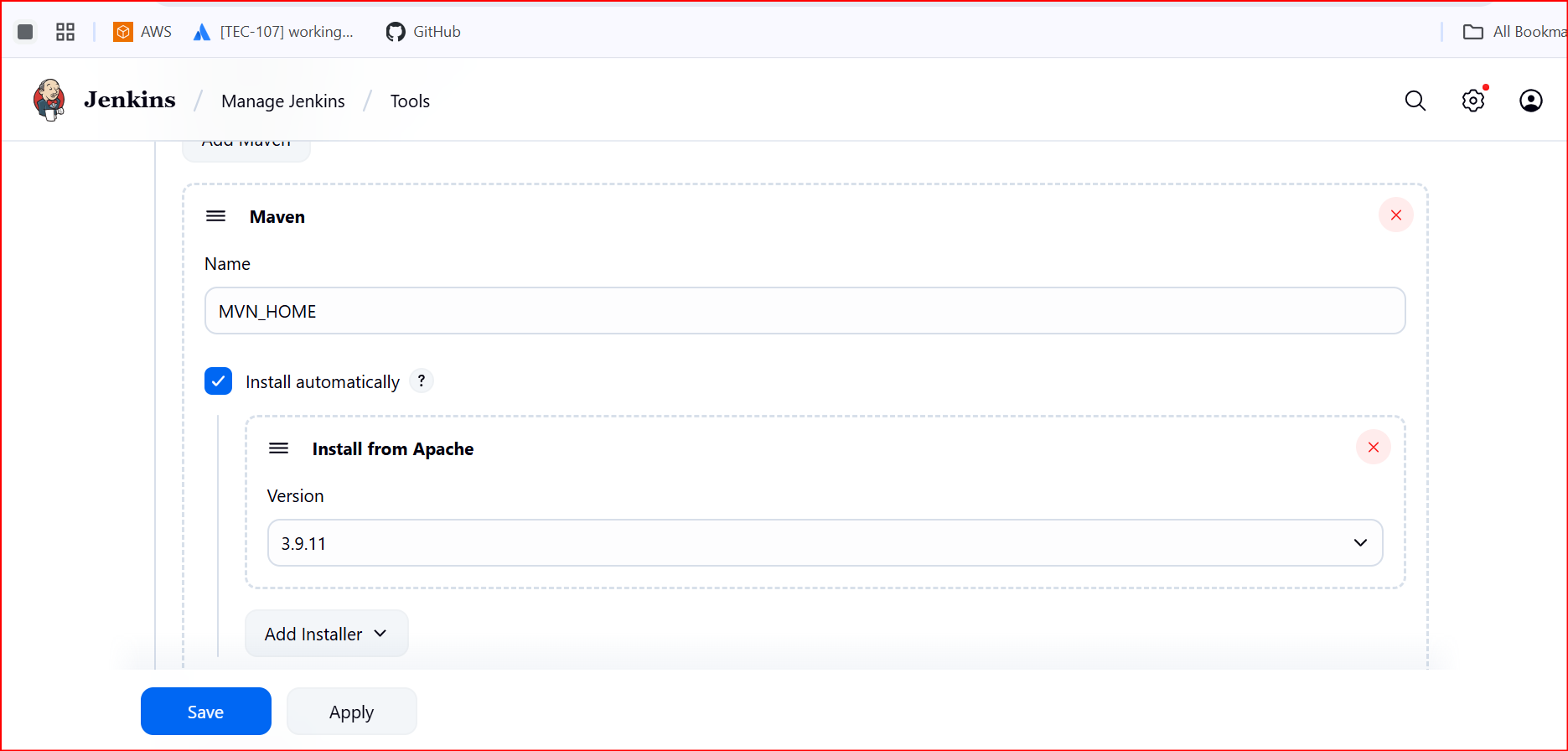
**Step 2: Configure SonarQube Server**

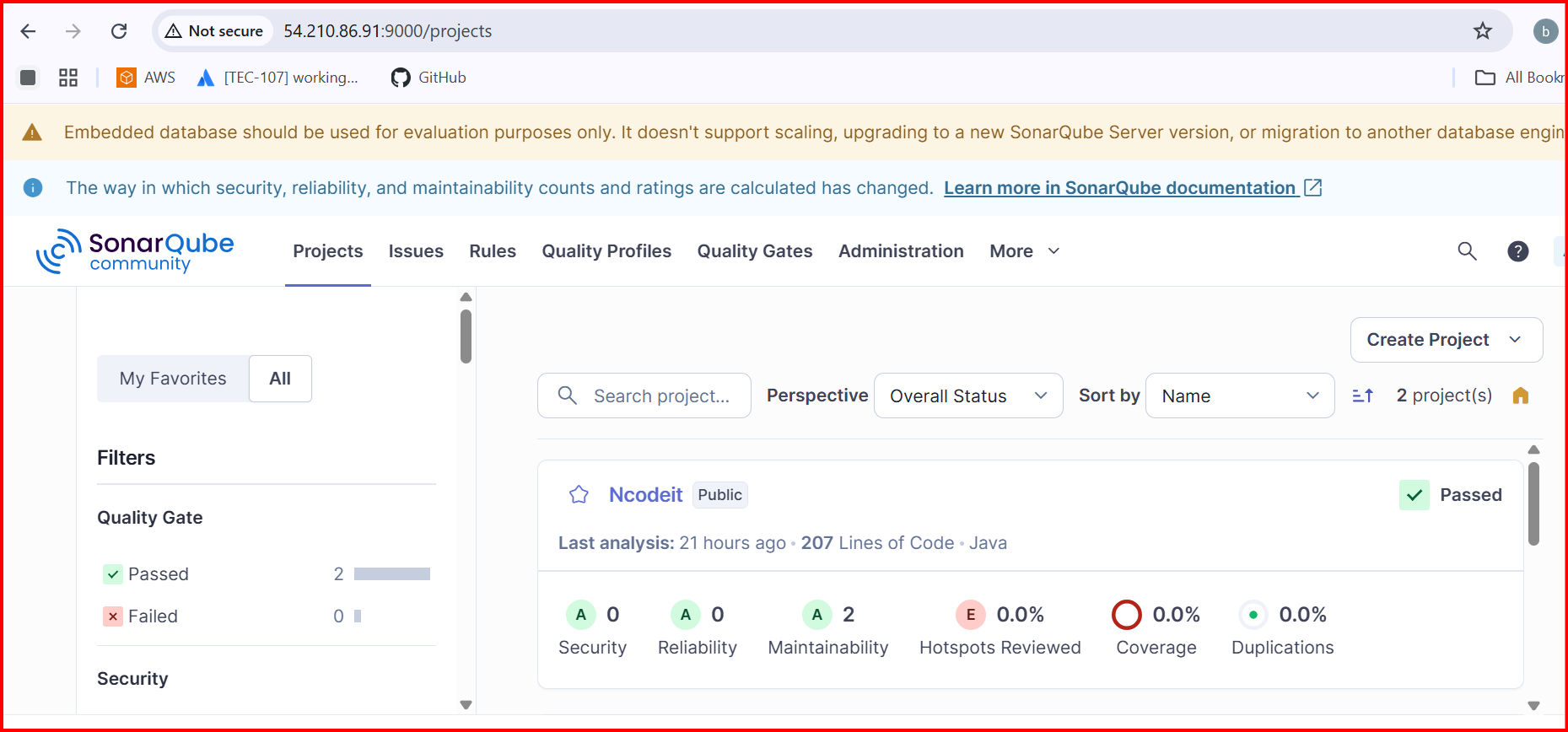
1. Go to **Manage Jenkins → Configure System**.
2. Scroll to **SonarQube servers**.
3. Click **Add SonarQube**.
4. Give it a **Name** (e.g. SonarQube).
5. Enter your **SonarQube Server URL** (e.g. http://<sonar-server>:9000).
6. Add **Server authentication token**:
   * Generate a token in SonarQube (My Account → Security → Generate Token).
   * Store it in Jenkins credentials (type: Secret Text).
   * Select it here.

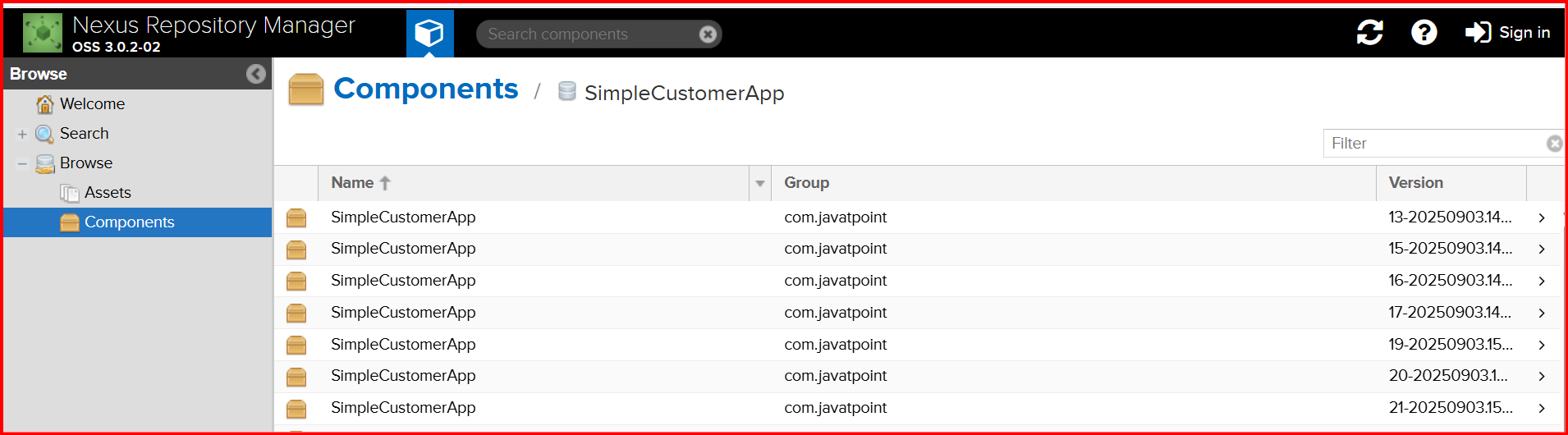


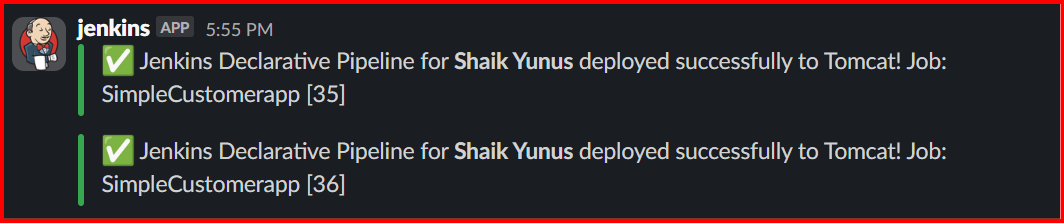
**Configure Maven in Jenkins**

1. Go to **Manage Jenkins → Tools**.
2. Scroll to **Maven installations**.
3. Click **Add Maven**.
4. Enter a **Name** → e.g. MVN\_HOME (this must match the Jenkinsfile).
5. Tick **Install automatically** → choose version (like 3.9.x).









**Tomcat deployment**

**yum install java-1.8\***

**wget https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz**

**tar xvf apache-tomcat-9.0.108.tar.gz**

**cd apache-tomcat-9.0.108/**

**cd bin/**

**ls**

**bash startup.sh**

**find / -name context.xml**

**4 files will be avalible we can edit the files file value there <!-- -->**

**[root@ip-172-31-40-12 apache-tomcat-9.0.108]# cd conf/**

**vi tomcat-users.xml**

**<role rolename="manager-gui"/>**

**<role rolename="manager-script"/>**

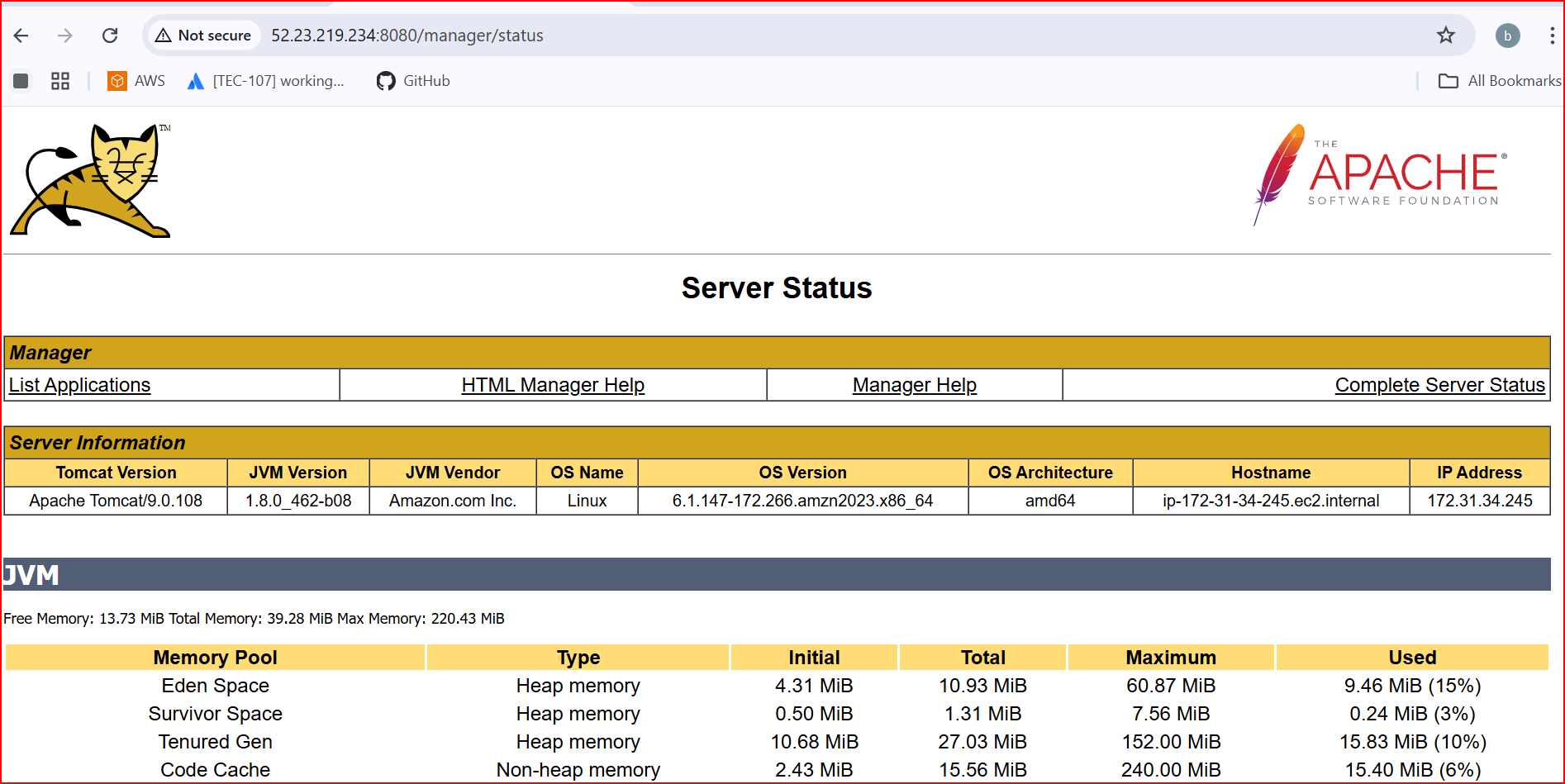
**<role rolename="manager-jmx"/>**

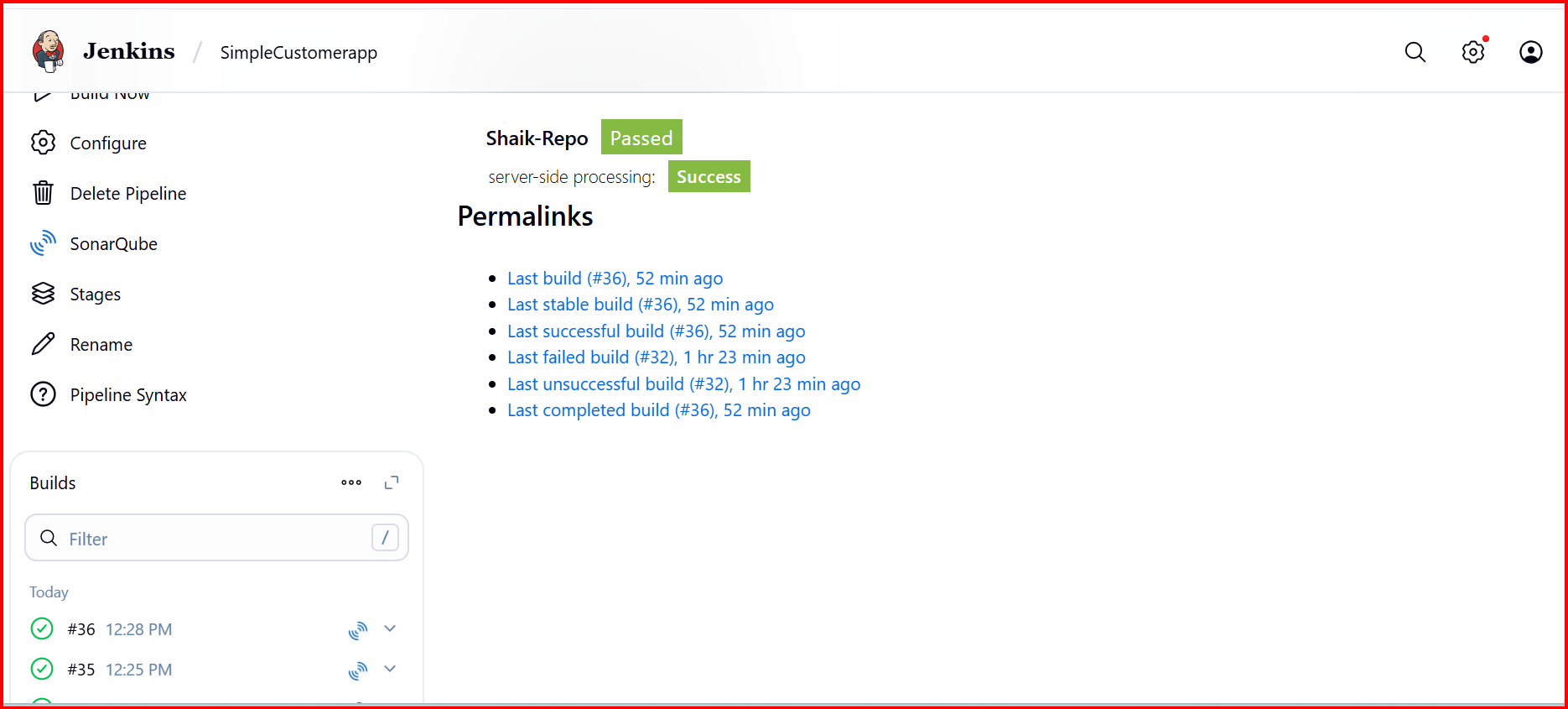
**<role rolename="manager-status"/>**

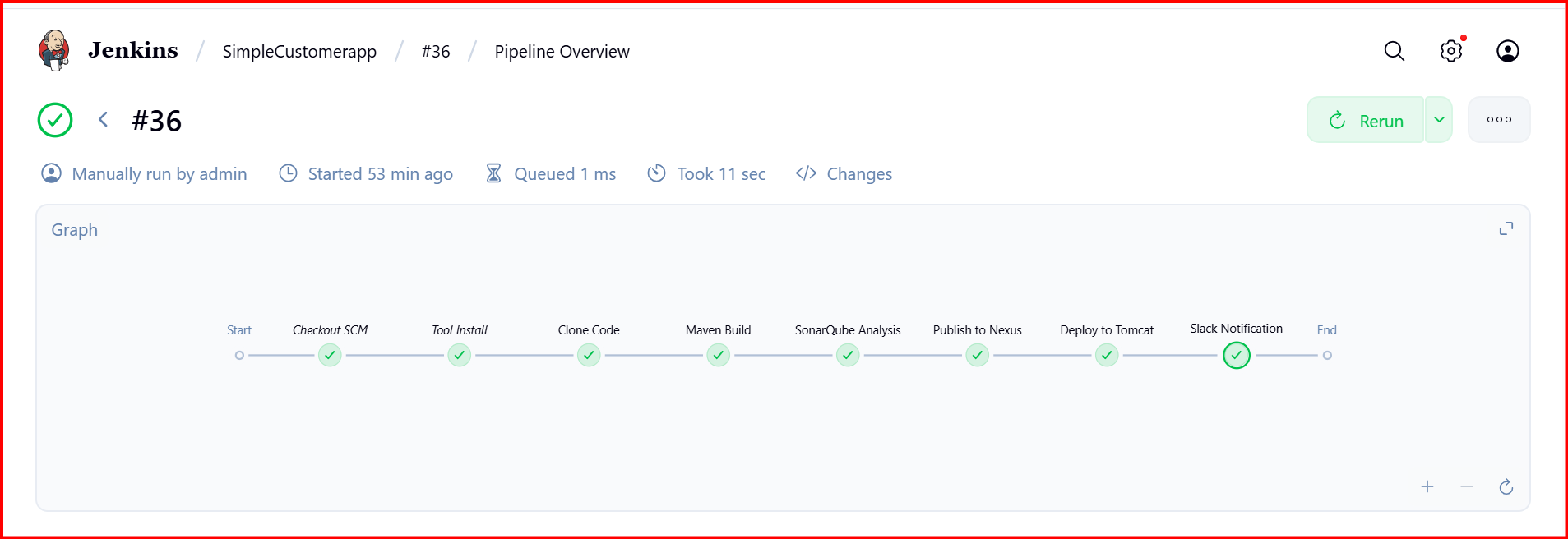
**<user username="admin" password="admin" roles="manager-gui, manager-script, manager-jmx, manager-status"/>**

**<user username="deployer" password="deployer" roles="manager-script"/>**

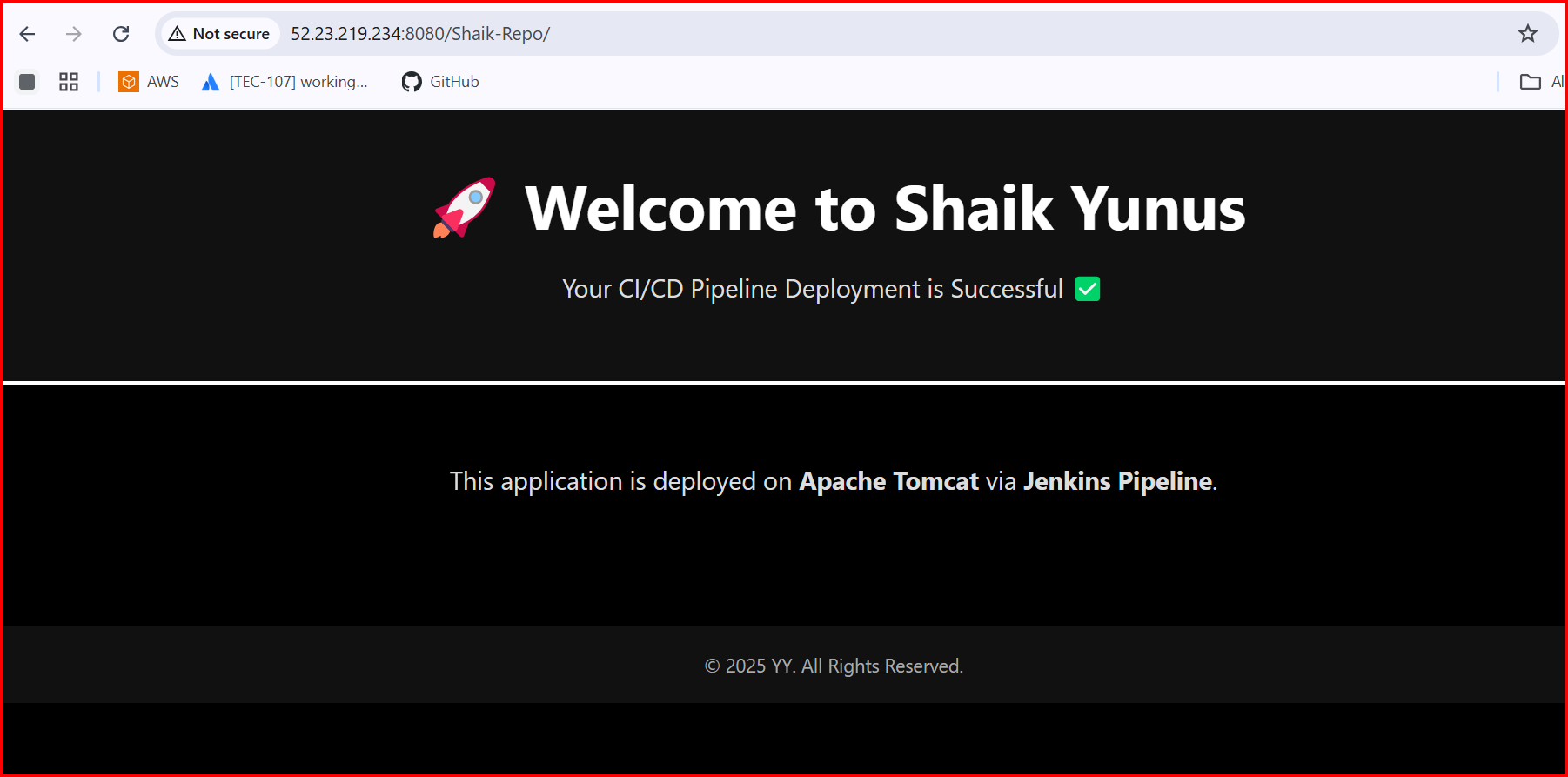
**<user username="tomcat" password="s3cret" roles="manager-gui"/>**

****

****

****

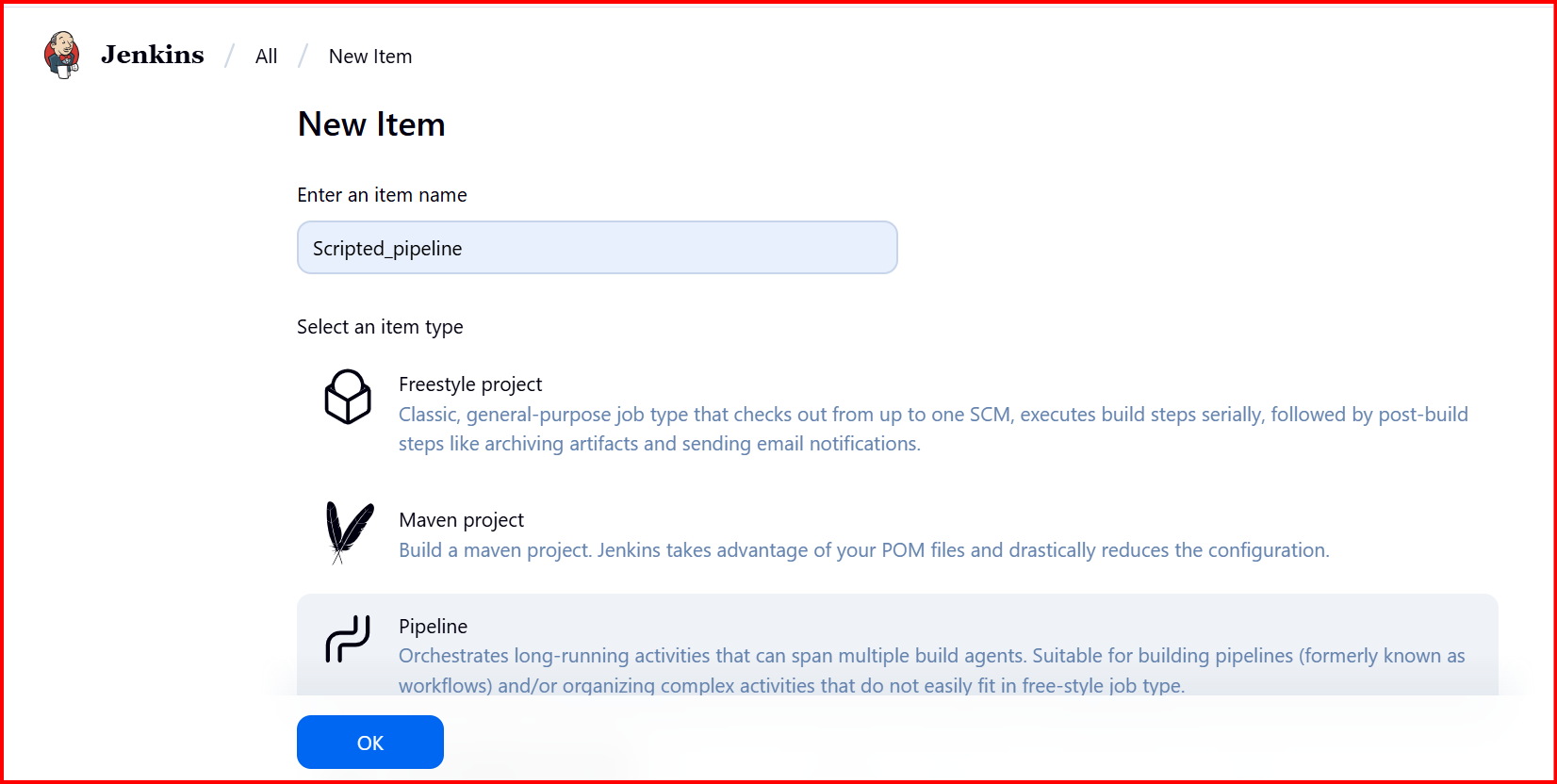
[**http://52.23.219.234:8080/Shaik-Repo/**](http://52.23.219.234:8080/Shaik-Repo/)

****

1. **Create one Scripted pipeline job**

**Jenkins dashboard → New Item.**

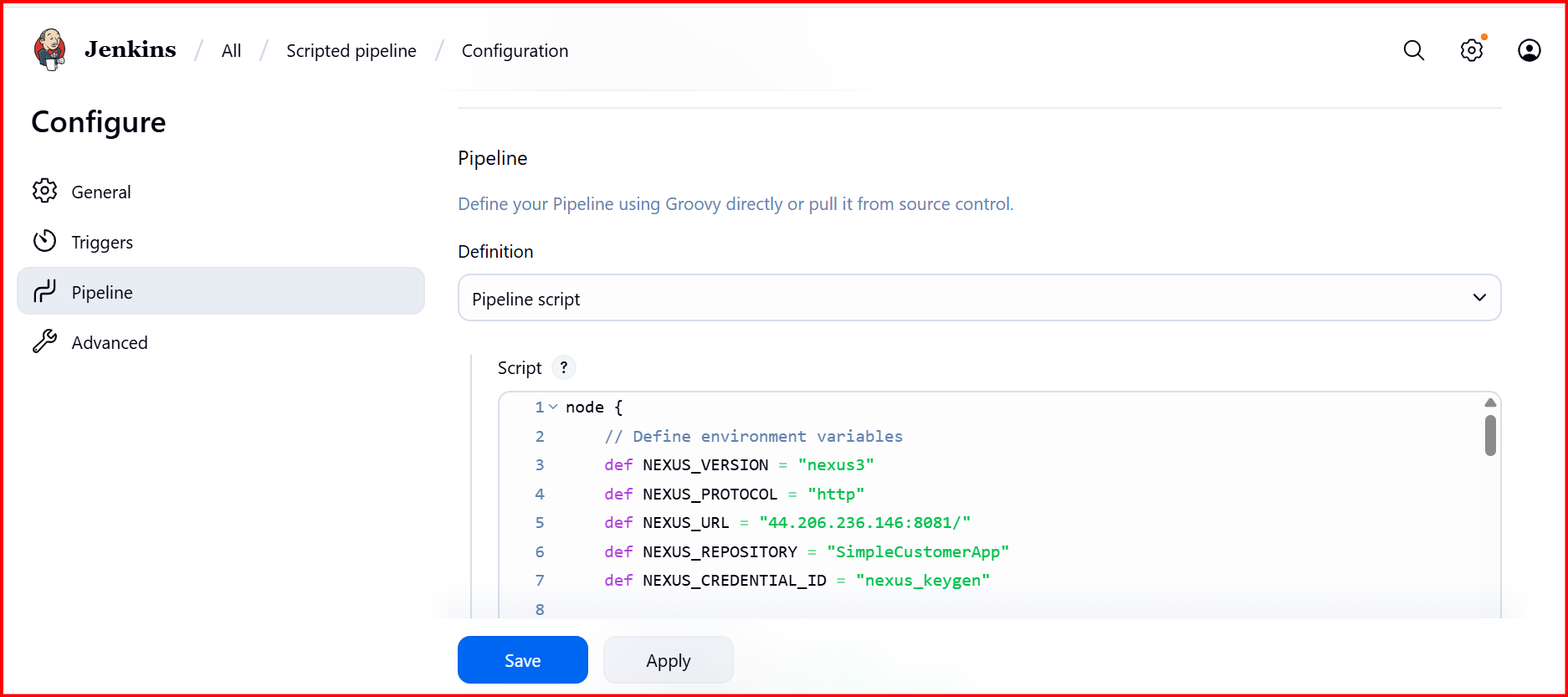
* **Name: Scripted-Pipeline-Job (example)**
* **Select Pipeline and click OK.**

****

**Go to job configuration**

**Pipeline select Pipeline Script**

**Past your script**

****

**node {**

**// Define environment variables**

**def NEXUS\_VERSION = "nexus3"**

**def NEXUS\_PROTOCOL = "http"**

**def NEXUS\_URL = "44.206.236.146:8081/"**

**def NEXUS\_REPOSITORY = "SimpleCustomerApp"**

**def NEXUS\_CREDENTIAL\_ID = "nexus\_keygen"**

**// Define tools**

**def mvnHome = tool 'MVN\_HOME'**

**def scannerHome = tool 'sonar\_scanner'**

**try {**

**stage('Clone Code') {**

**git 'https://github.com/Shaik123-hu/sabear\_simplecutomerapp.git'**

**}**

**stage('Maven Build') {**

**sh "${mvnHome}/bin/mvn -Dmaven.test.failure.ignore=true clean install"**

**}**

**stage('SonarCloud') {**

**withSonarQubeEnv('sonarqube\_server') {**

**// Define directories**

**def sourceDir = "${WORKSPACE}/src/main/java"**

**def webappDir = "${WORKSPACE}/src/main/webapp"**

**def binaryDir = "${WORKSPACE}/target/classes"**

**def reportDir = "${WORKSPACE}/target/surefire-reports"**

**// Determine source directories to analyze**

**def sonarSources = []**

**if (fileExists(sourceDir)) {**

**sonarSources.add(sourceDir)**

**}**

**if (fileExists(webappDir)) {**

**sonarSources.add(webappDir)**

**}**

**if (sonarSources.isEmpty()) {**

**echo "Warning: No source directories (src/main/java or src/main/webapp) found. Skipping SonarQube analysis."**

**return**

**}**

**// Build SonarQube command**

**def sonarCommand = """**

**${scannerHome}/bin/sonar-scanner \**

**-Dsonar.projectKey=Ncodeit \**

**-Dsonar.projectName=Ncodeit \**

**-Dsonar.projectVersion=2.0 \**

**-Dsonar.sources="${sonarSources.join(',')}" \**

**-Dsonar.junit.reportsPath="${reportDir}" \**

**-Dsonar.verbose=true**

**"""**

**// Add binaries only if target/classes exists**

**if (fileExists(binaryDir)) {**

**sonarCommand += " -Dsonar.java.binaries=\"${binaryDir}\""**

**} else {**

**echo "Warning: Binary directory ${binaryDir} does not exist. Skipping Java binaries analysis."**

**}**

**// Execute SonarQube analysis**

**sh sonarCommand**

**}**

**}**

**stage('Publish to Nexus') {**

**def pom = readMavenPom file: 'pom.xml'**

**def groupId = pom.groupId**

**def artifactId = pom.artifactId**

**// Use a fallback version if ${BUILD\_NUMBER} causes issues**

**def version = pom.version.replace('${BUILD\_NUMBER}', env.BUILD\_NUMBER ?: '3')**

**def packaging = pom.packaging**

**def filesByGlob = findFiles(glob: "target/\*.${packaging}")**

**if (filesByGlob.length == 0) {**

**error "\*\*\* No artifact found in target/\*.${packaging}"**

**}**

**def artifactPath = filesByGlob[0].path**

**if (fileExists(artifactPath)) {**

**echo "\*\*\* Uploading ${artifactPath} to Nexus (group: ${groupId}, version: ${version}, packaging: ${packaging})"**

**nexusArtifactUploader(**

**nexusVersion: NEXUS\_VERSION,**

**protocol: NEXUS\_PROTOCOL,**

**nexusUrl: NEXUS\_URL,**

**groupId: groupId,**

**version: version,**

**repository: NEXUS\_REPOSITORY,**

**credentialsId: NEXUS\_CREDENTIAL\_ID,**

**artifacts: [**

**[artifactId: artifactId, classifier: '', file: artifactPath, type: packaging],**

**[artifactId: artifactId, classifier: '', file: 'pom.xml', type: 'pom']**

**]**

**)**

**} else {**

**error "\*\*\* File: ${artifactPath}, could not be found"**

**}**

**}**

**stage('Deploy to Tomcat') {**

**withCredentials([usernamePassword(credentialsId: 'tomcat', usernameVariable: 'TOMCAT\_USER', passwordVariable: 'TOMCAT\_PASS')]) {**

**def warFile = sh(script: "ls target/\*.war | head -n 1", returnStdout: true).trim()**

**echo "Deploying ${warFile} to Tomcat at context path /SimpleCustomerApp..."**

**sh """**

**curl -u ${TOMCAT\_USER}:${TOMCAT\_PASS} \**

**-T ${warFile} \**

**"http://52.23.219.234:8080/manager/text/deploy?path=/SimpleCustomerApp&update=true"**

**"""**

**}**

**}**

**// Success notification**

**slackSend(**

**channel: '#jenkins-integration',**

**color: 'good',**

**message: "Scripted pipeline for Simple Customer App has been successfully deployed in Tomcat :white\_check\_mark: by Yunus for Job{env.JOB\_NAME} [${env.BUILD\_NUMBER}]",**

**tokenCredentialId: 'Slackid'**

**)**

**} catch (e) {**

**// Failure notification**

**slackSend(**

**channel: '#jenkins-integration',**

**color: 'danger',**

**message: "❌ Build/Deploy Failed: ${env.JOB\_NAME} [${env.BUILD\_NUMBER}]",**

**tokenCredentialId: 'Slackid'**

**)**

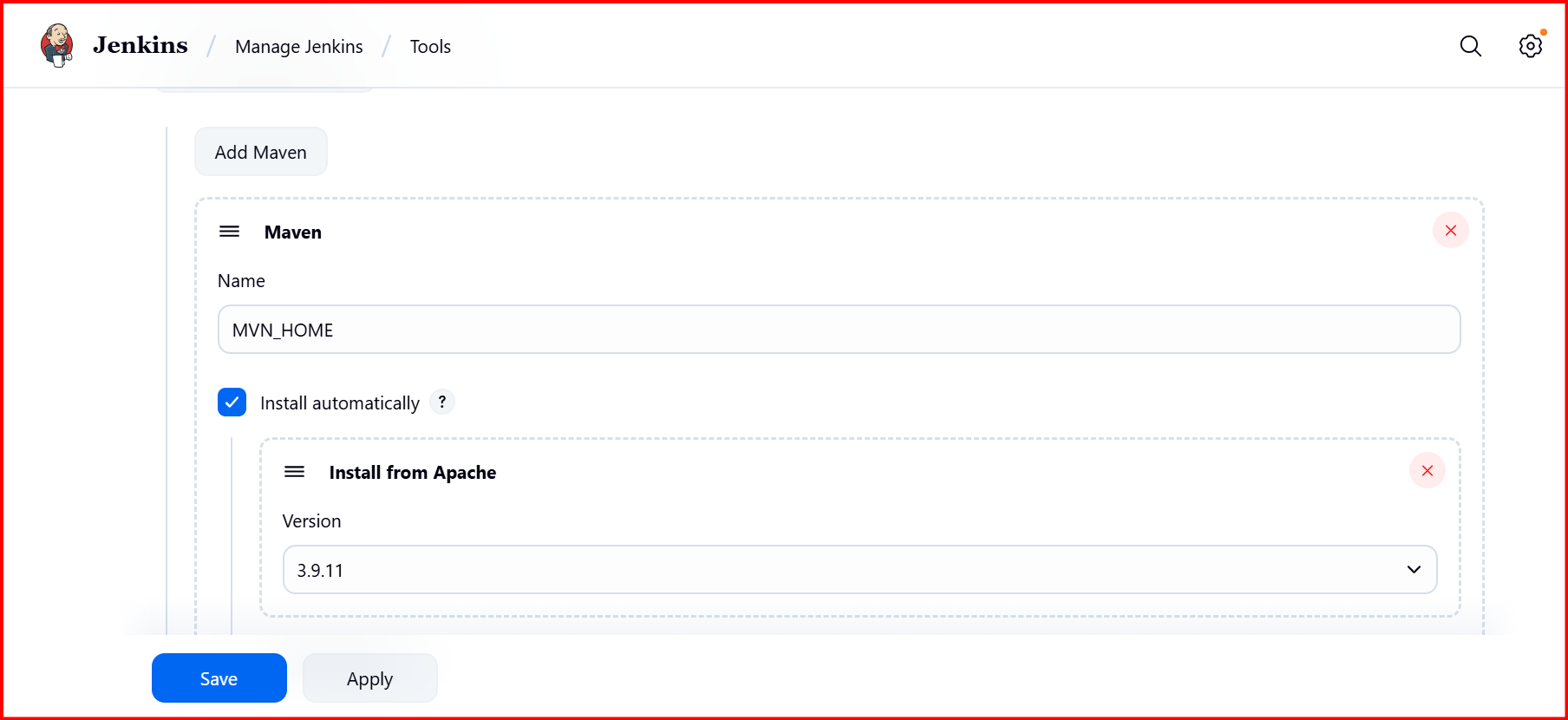
**throw e**

**}**

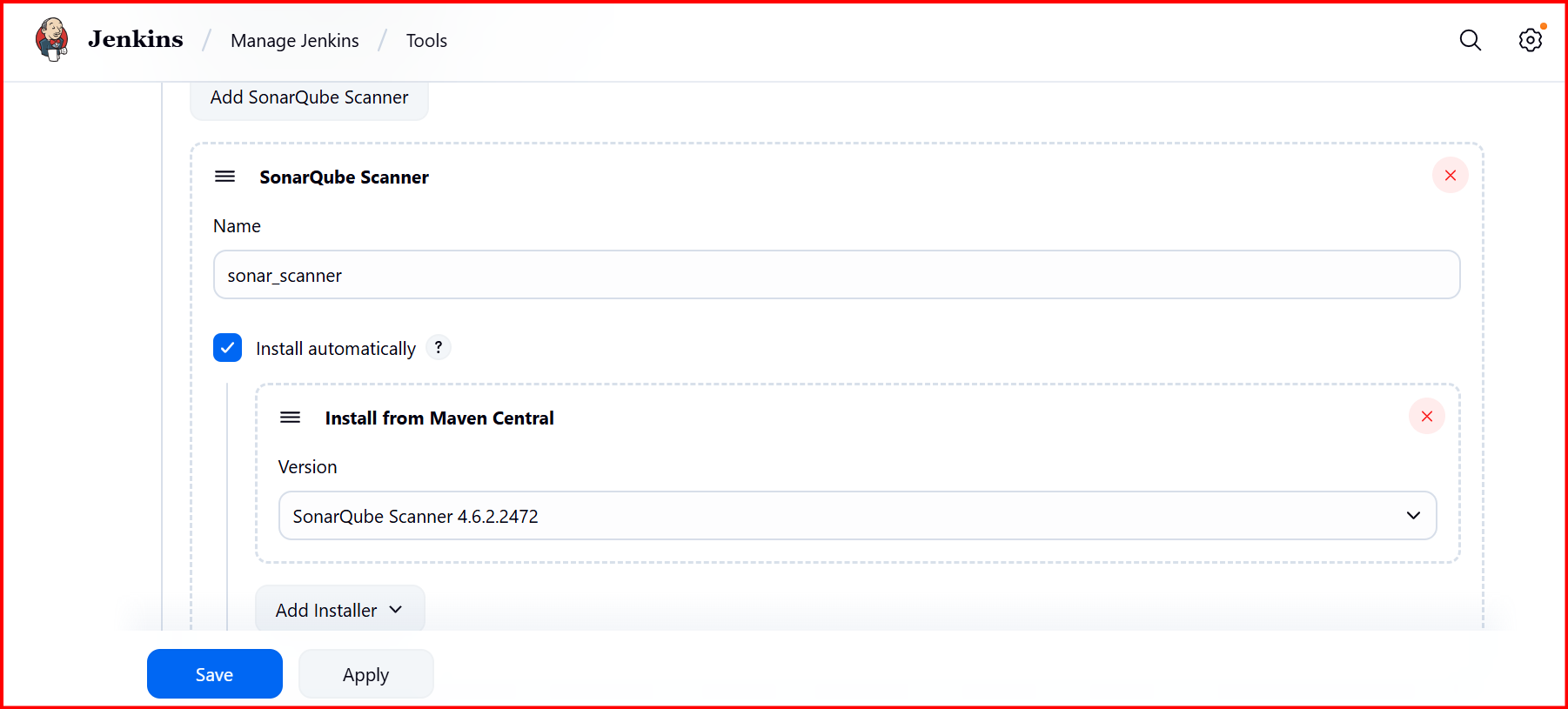
**}**

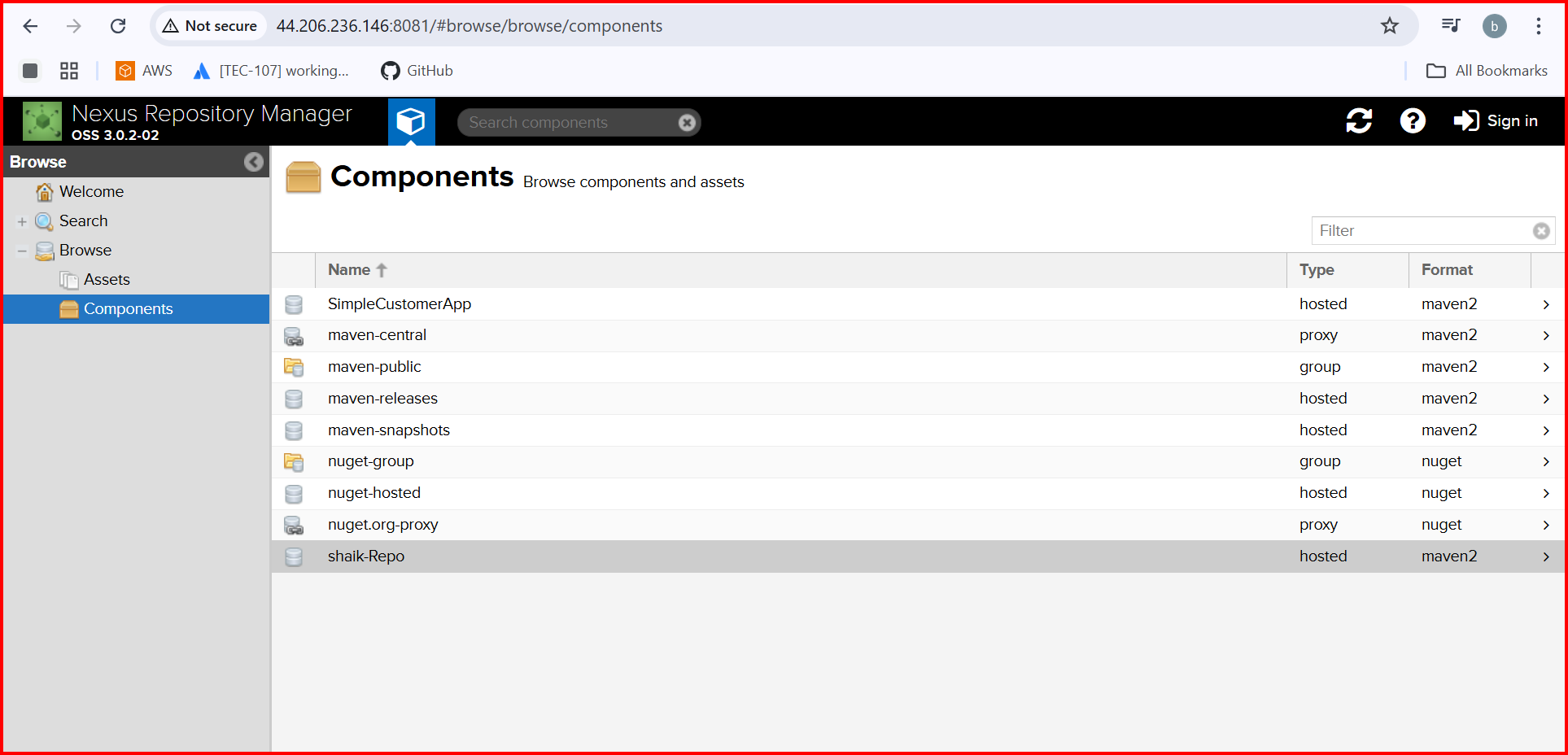
**Manage Jenkins → Global Tool Configuration:**

* **Add Maven installation → Name it (e.g. MAVEN) → Jenkins will auto-install or you can put path.**

****

**Manage Jenkins → Configure System → SonarQube servers: add MySonarServer and token (or configure under Global Tool Config if using scanner).**

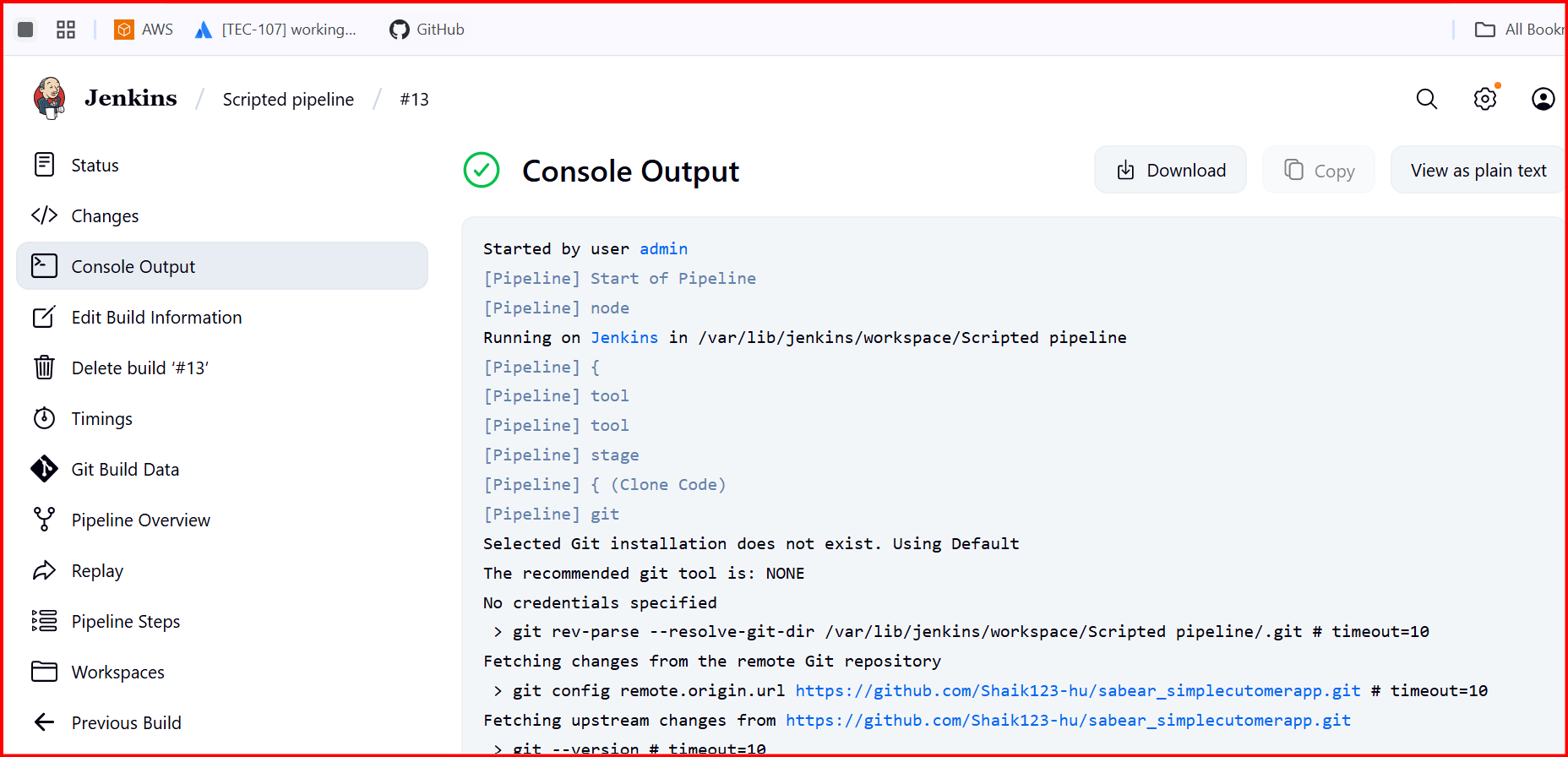
****

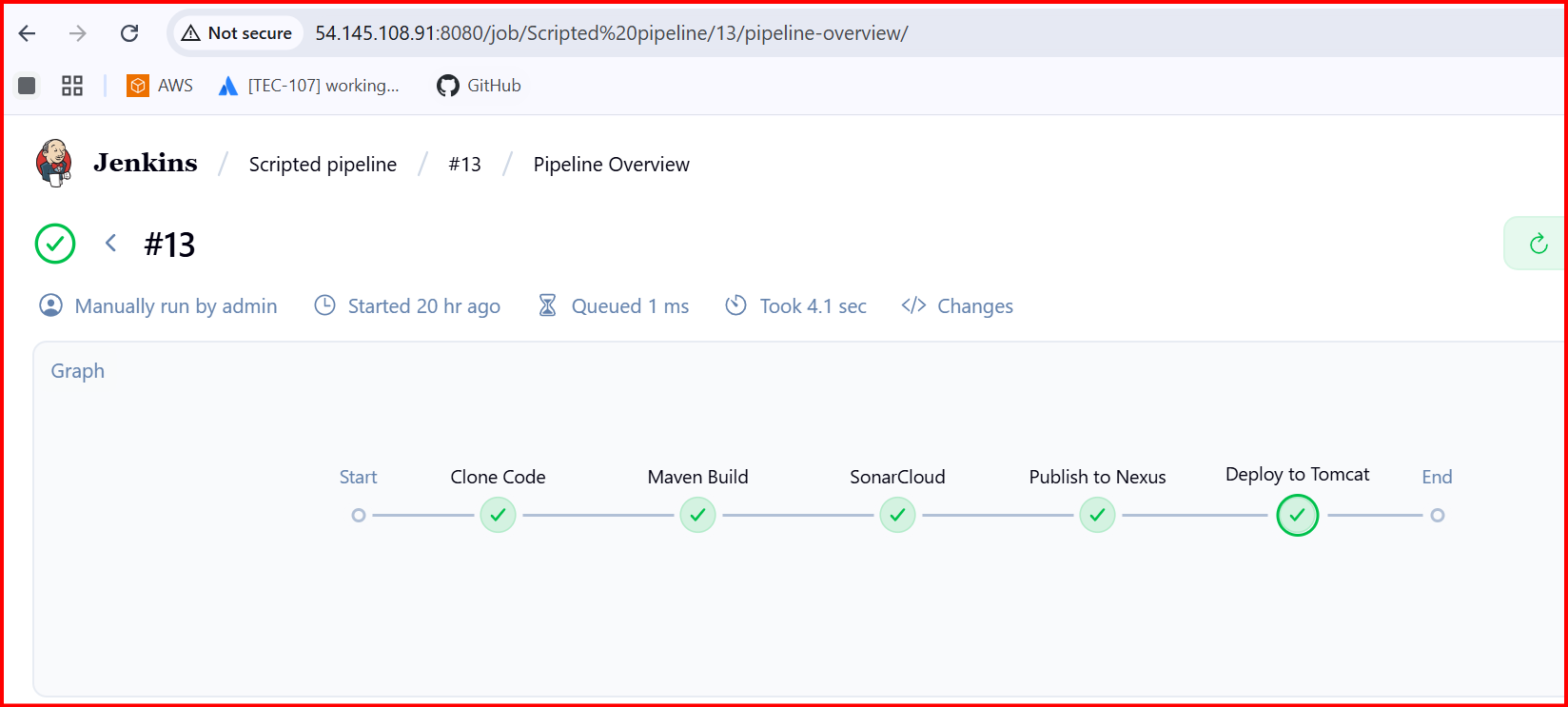
****

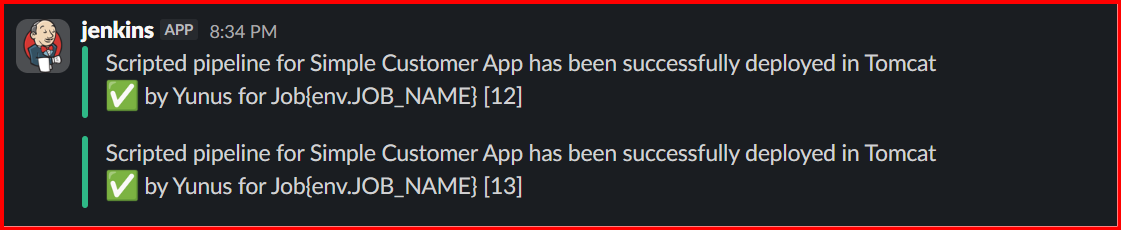
**Configure Slack plugin (Manage Jenkins → Configure System → Slack) or use slackSend with tokenCredentialId in pipeline.**

**Save & run**

* **Save the job, click Build Now.**
* **Open the build → Console Output to follow each stage.**

****

****

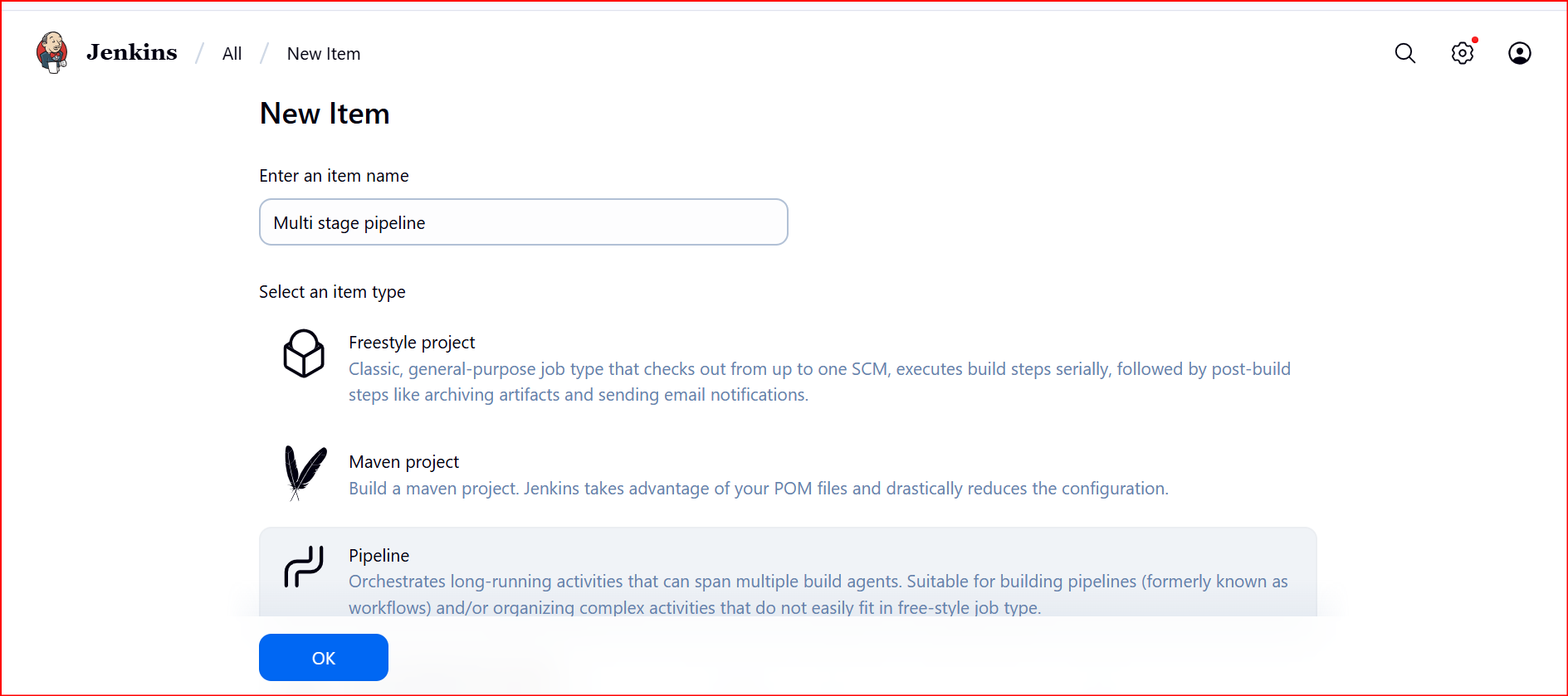
****

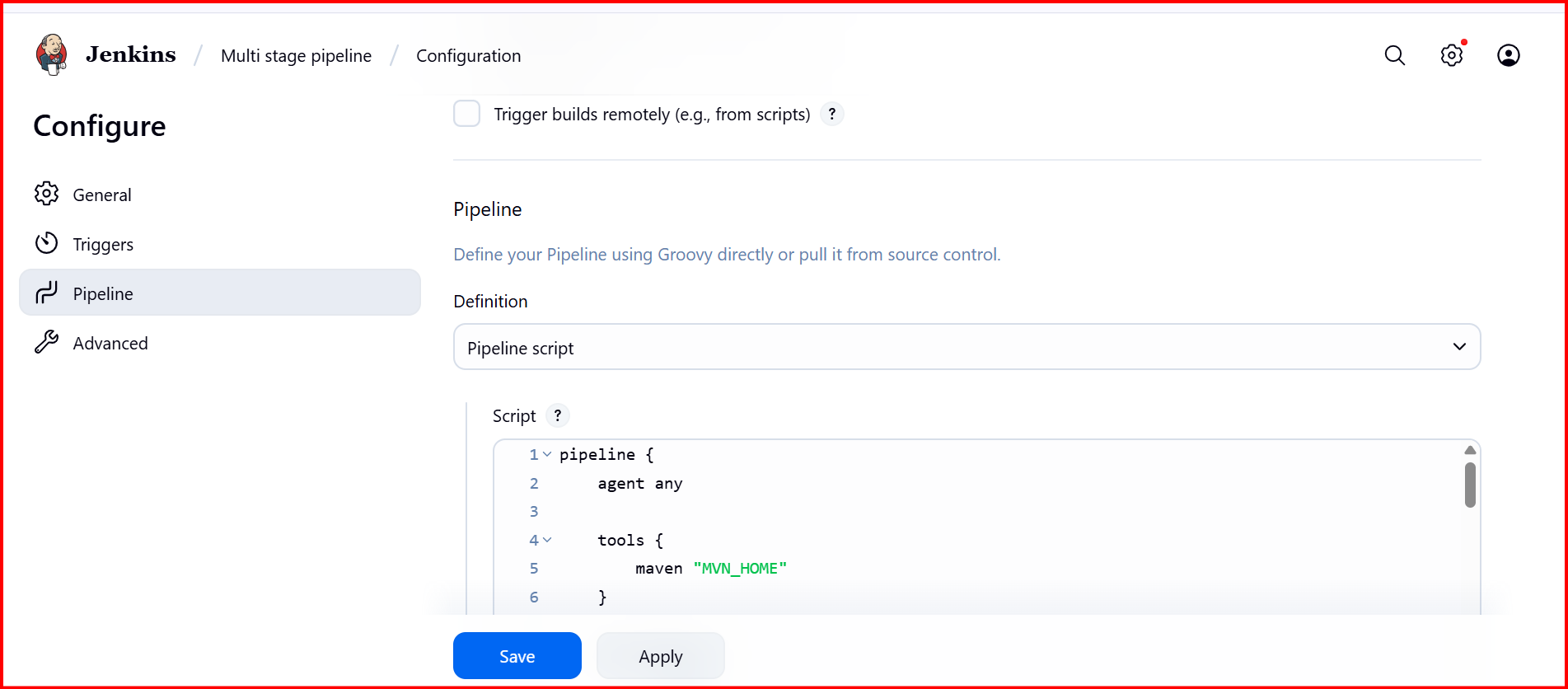
**3.Create one multi stage pipeline job**

**A multi-stage pipeline means your Jenkins job is divided into sequential stages (steps of the build lifecycle).  
Each stage has a specific purpose, and Jenkins executes them one after another.**

**A multi-stage pipeline = Jenkins pipeline with multiple stages that automate the entire software delivery process (from code → build → test → deploy → notify).**

**Create Multi stage pipeline In New Iteams**

****

****

**pipeline {**

**agent any**

**tools {**

**maven "MVN\_HOME"**

**}**

**environment {**

**// Nexus**

**NEXUS\_URL = "44.206.236.146:8081"**

**NEXUS\_REPOSITORY = "shaik\_Repo"**

**NEXUS\_CREDENTIAL\_ID = "nexus\_keygen"**

**// Tomcat**

**TOMCAT\_USER = "deployer"**

**TOMCAT\_PASSWORD = "deployer"**

**TOMCAT\_HOST = "52.23.219.234"**

**TOMCAT\_PORT = "8080"**

**// Slack**

**SLACK\_CHANNEL = "#jenkins-integration"**

**SLACK\_CREDENTIAL\_ID = "slack\_notification"**

**}**

**stages {**

**stage("Clone Code") {**

**steps {**

**git url: 'https://github.com/Shaik123-hu/spring3-mvc-maven-xml-hello-world-1.git'**

**}**

**}**

**stage("Maven Build") {**

**steps {**

**sh 'mvn -B -Dmaven.test.failure.ignore=true clean install'**

**}**

**}**

**stage("Publish to Nexus") {**

**steps {**

**script {**

**def pom = readMavenPom file: 'pom.xml'**

**def artifactVersion = pom.version**

**def groupId = pom.groupId**

**def artifactId = pom.artifactId**

**def warFiles = findFiles(glob: "target/${artifactId}-${artifactVersion}.war")**

**if (warFiles.length == 0) {**

**error "WAR file not found: target/${artifactId}-${artifactVersion}.war"**

**}**

**def warFile = warFiles[0].path**

**echo "📦 Uploading ${warFile} to Nexus Repository: ${NEXUS\_REPOSITORY}"**

**nexusArtifactUploader(**

**artifacts: [[**

**artifactId: artifactId,**

**classifier: '',**

**file: warFile,**

**type: 'war'**

**], [**

**artifactId: artifactId,**

**classifier: '',**

**file: 'pom.xml',**

**type: 'pom'**

**]],**

**credentialsId: NEXUS\_CREDENTIAL\_ID,**

**groupId: groupId,**

**version: artifactVersion,**

**repository: NEXUS\_REPOSITORY**

**)**

**}**

**}**

**}**

**stage("Deploy to Tomcat") {**

**steps {**

**script {**

**def pom = readMavenPom file: 'pom.xml'**

**def artifactVersion = pom.version**

**def artifactId = pom.artifactId**

**def warFile = "target/${artifactId}-${artifactVersion}.war"**

**echo "🚀 Deploying ${warFile} to Tomcat at ${TOMCAT\_HOST}:${TOMCAT\_PORT}"**

**sh """**

**curl -u ${TOMCAT\_USER}:${TOMCAT\_PASSWORD} \**

**-T ${warFile} \**

**"http://${TOMCAT\_HOST}:${TOMCAT\_PORT}/manager/text/deploy?path=/${artifactId}&update=true"**

**"""**

**}**

**}**

**}**

**}**

**post {**

**success {**

**slackSend(**

**channel: SLACK\_CHANNEL,**

**color: 'good',**

**message: "✅ Pipeline '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]' completed successfully! <${env.BUILD\_URL}|Open Build>"**

**)**

**}**

**failure {**

**slackSend(**

**channel: SLACK\_CHANNEL,**

**color: 'danger',**

**message: "❌ Pipeline '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]' failed! <${env.BUILD\_URL}|Open Build>"**

**)**

**}**

**always {**

**echo "🧹 Cleaning workspace..."**

**cleanWs()**

**}**

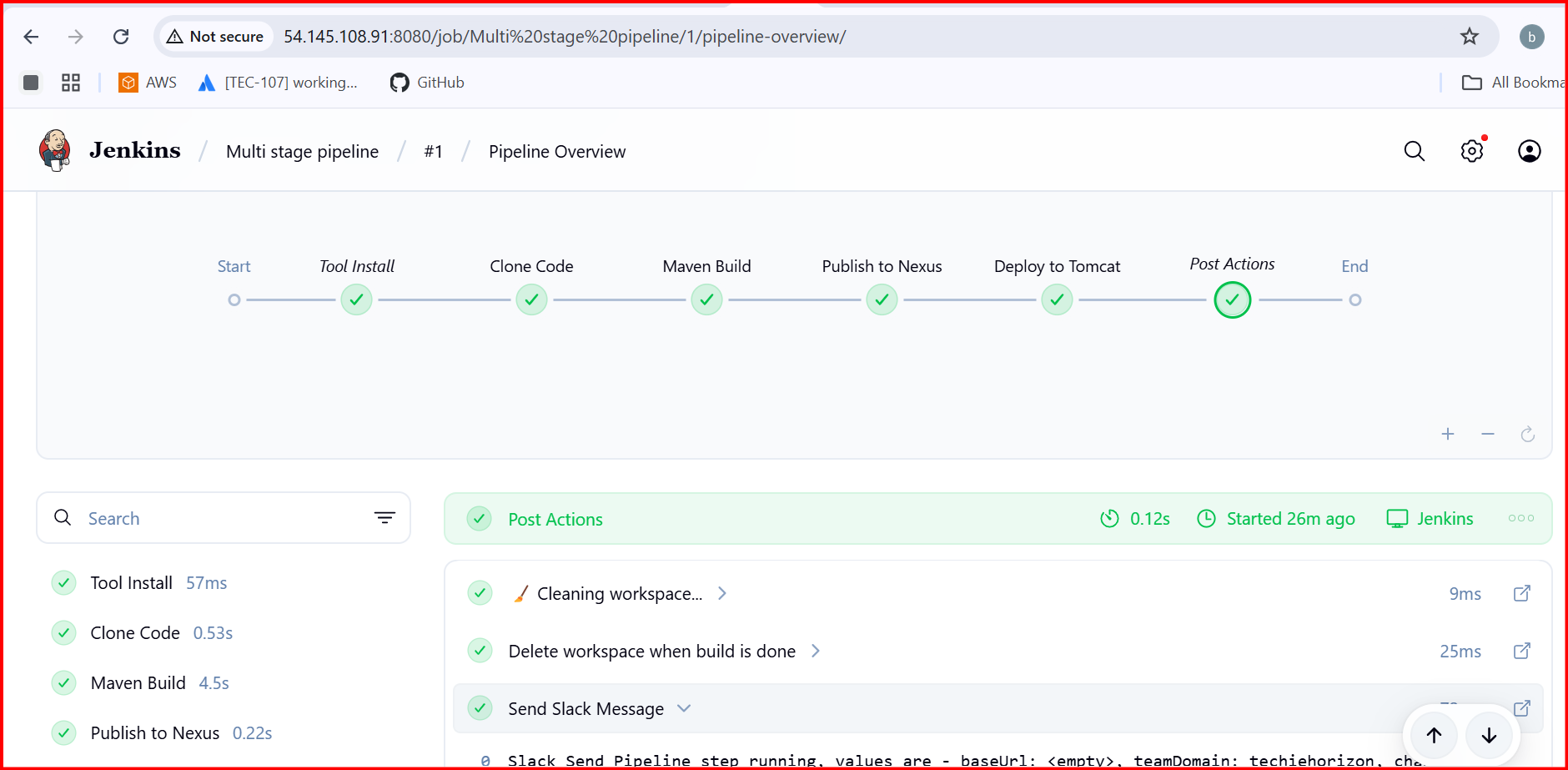
**}**

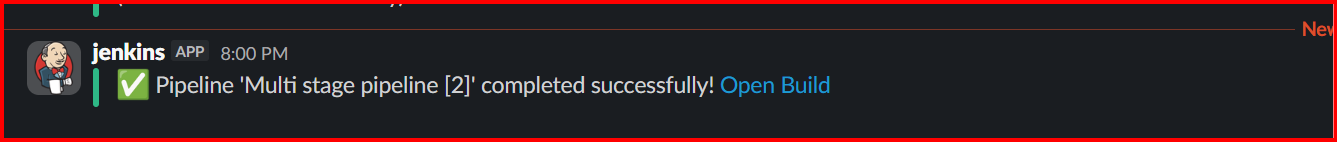
**}**

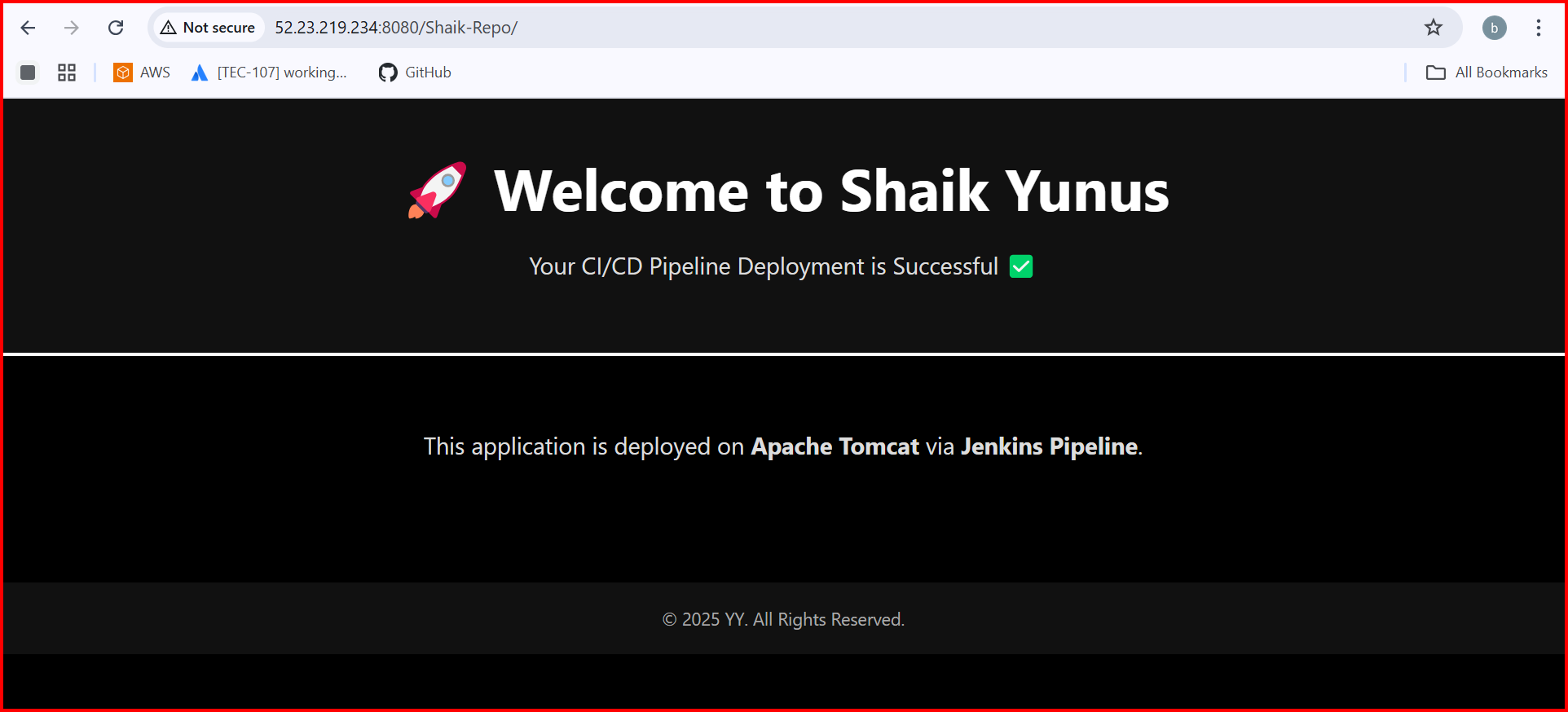
**Save & Build**

* **Click Save.**
* **Run Build Now.**
* **You’ll see multiple stages in Jenkins UI (Git Clone → Build → Code Analysis → Test → Package → Deploy).**

****

****

****

****

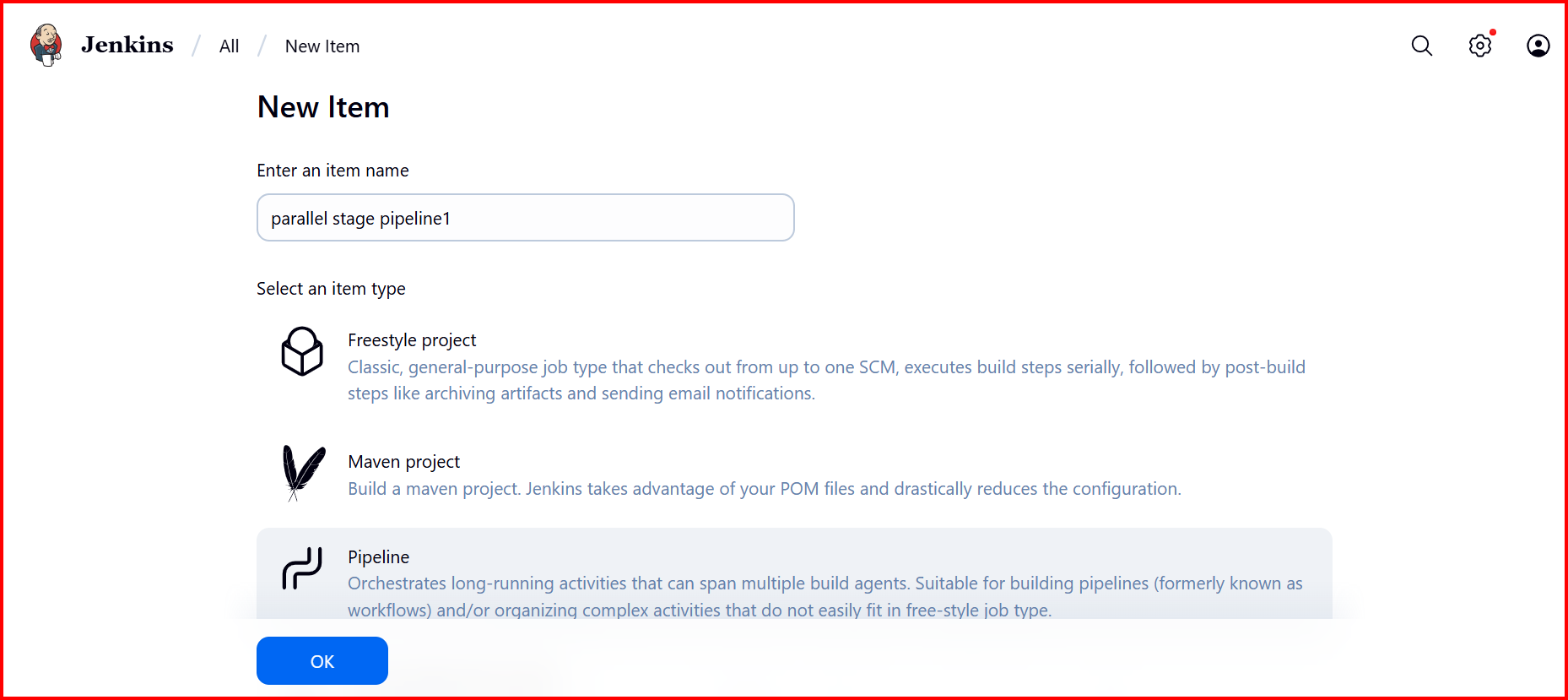
**4.Create one parallel stage pipeline job**

**Step 1: Access Jenkins**

* **Open your Jenkins URL in a browser: http://<jenkins-server>:8080.**
* **Sign in with your Jenkins username and password.**

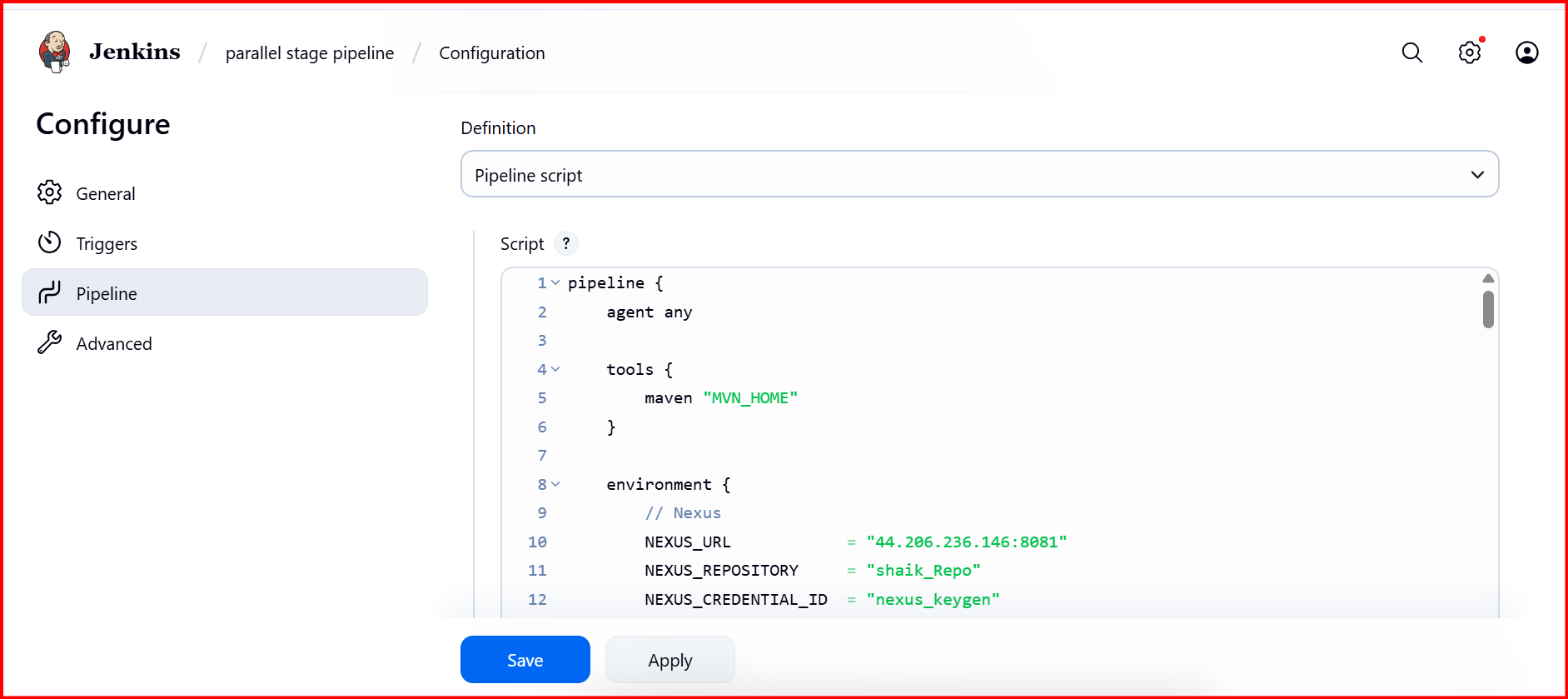
**Step 2: Start a New Job**

* **From the Jenkins dashboard, click on “New Item”.**
* **Type a name for your job (for example: Parallel-Pipeline).**
* **Choose Pipeline as the project type.**
* **Click OK to create the job.**

****

**Configure the Pipeline**

* **Scroll to the Pipeline section.**
* **Choose one of these:**
  + **Pipeline script → Paste your pipeline code directly.**
  + **Pipeline script from SCM → Fetch pipeline code (Jenkinsfile) from Git.**

****

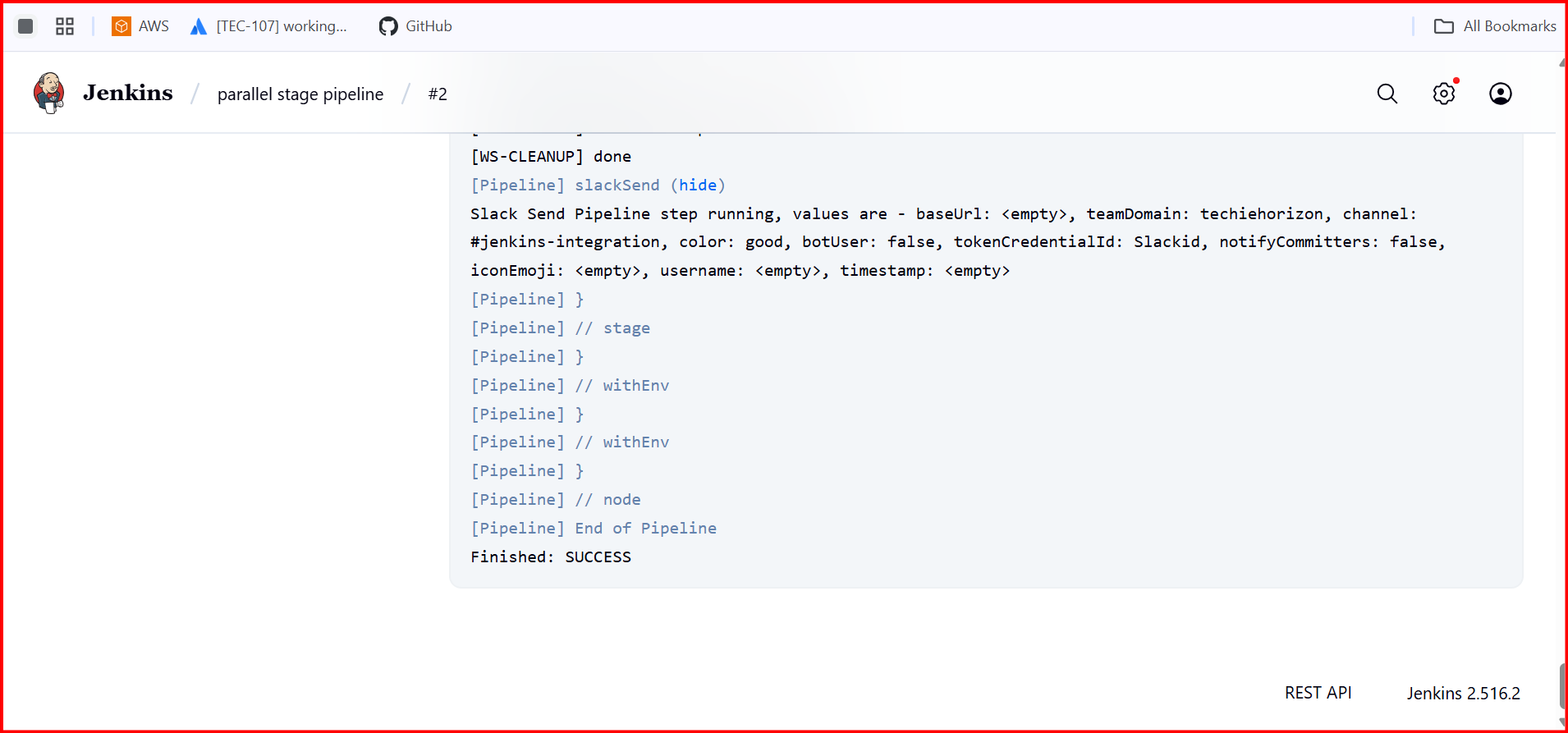
**Define Parallel Stages**

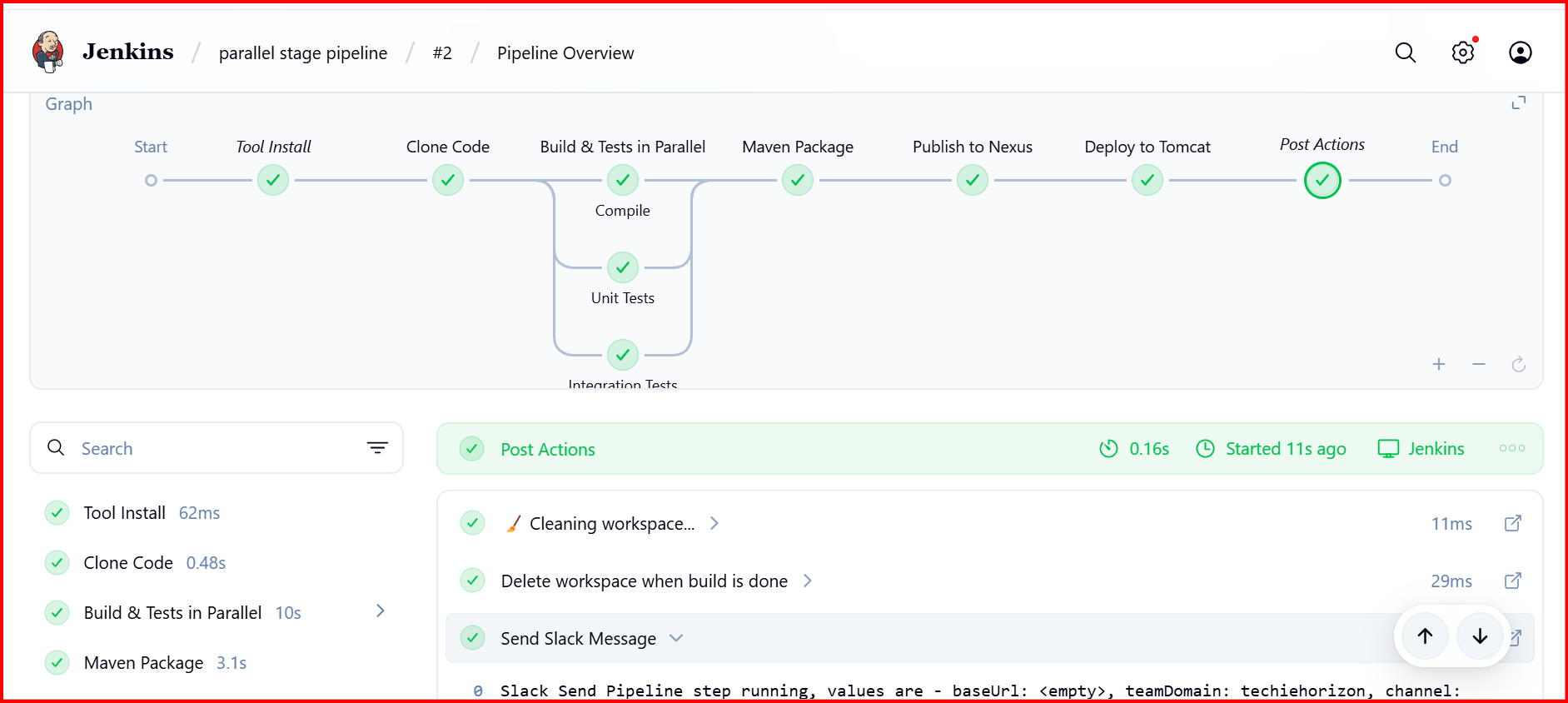
* **Inside your pipeline, add a parallel {} block under a stage.**
* **Each branch inside parallel {} runs at the same time.**

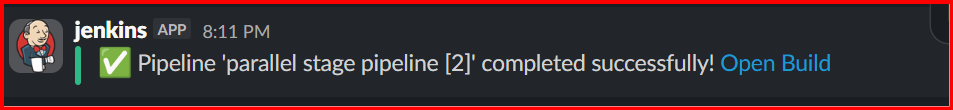
**Save and Build**

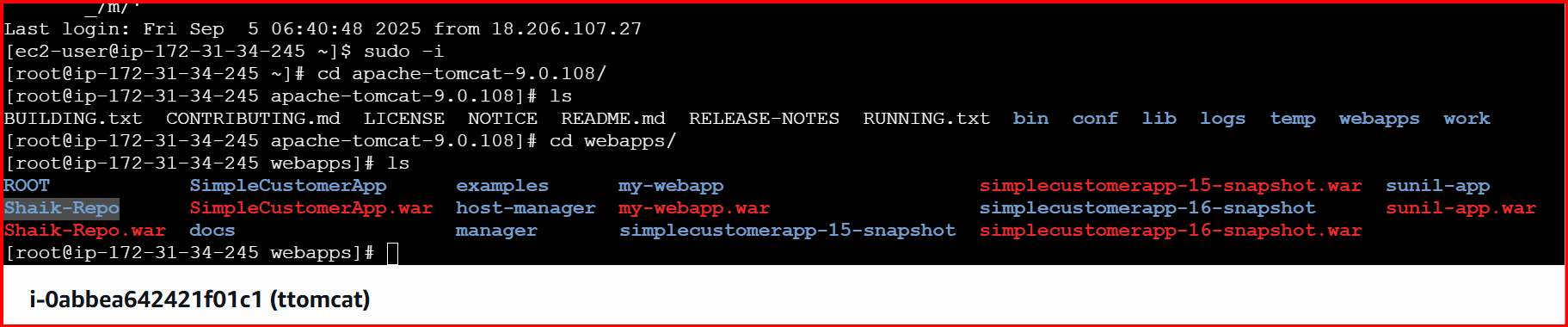
* **Click Save.**

**Click Build Now**

****

****

****

****

| **Feature / Aspect** | **Multi-Stage Pipeline** | **Parallel Stage Pipeline** |
| --- | --- | --- |
| **Definition** | **A pipeline broken into sequential stages (one after another).** | **A pipeline where multiple stages run at the same time (concurrently).** |
| **Execution Flow** | **Stages execute step by step in order. The next stage starts only after the previous one completes.** | **Multiple branches of stages are executed simultaneously.** |
| **Use Case** | **Best for workflows like Build → Test → Deploy.** | **Best for running different types of tests (unit, integration, functional) or builds on multiple environments (Java 8, Java 11, etc.) at the same time.** |

**Multi-Stage = sequential, step-by-step pipeline.**

**Parallel Stage = concurrent, side-by-side execution for speed and efficiency.**