**JENKINS Assignment-04.**

1) Setup jenkins CICD pipeline using freestyle job using Docker containers using below code.

<https://github.com/betawins/hiring-app.git>

Stages:

1) Git Clone

2) Sonarqube Integration

3) Maven Compilation

4) Nexus Artifactory

5) Slack Notification

6) Deploy On tomcat

2) Setup a jenkins CICD pipeline using Declarative pipeline using feature-1.1 branch.

<https://github.com/betawins/sabear_simplecutomerapp/tree/feature-1.1>

stages:

1) Git Clone

2) Sonarqube Integration

3) Maven Compilation

4) Nexus Artifactory

5) Slack Notification

6) Deploy On tomcat

3) Setup a jenkins CICD pipeline using Scripted pipeline using feature-1.1 branch.

<https://github.com/betawins/sabear_simplecutomerapp/tree/feature-1.1>

stages:

1) Git Clone

2) Sonarqube Integration

3) Maven Compilation

4) Nexus Artifactory

5) Slack Notification

6) Deploy On tomcat

4) Write sample skeleton of pipelines.

5) Create a parametirized job in jenkins.

<https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git>

6) Setup one slave machine for jenkins.

[**GitHub - betawins/hiring-app**](https://github.com/betawins/hiring-app.git)

[**GitHub - betawins/sabear\_simplecutomerapp at feature-1.1**](https://github.com/betawins/sabear_simplecutomerapp/tree/feature-1.1)

[**GitHub - betawins/spring3-mvc-maven-xml-hello-world-1: Maven + Spring 3 MVC hello world example (XML)**](https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git)

**1) Setup jenkins CICD pipeline using freestyle job using Docker containers using below code.**

<https://github.com/betawins/hiring-app.git>

Stages:

1) Git Clone

2) Sonarqube Integration

3) Maven Compilation

4) Nexus Artifactory

5) Slack Notification

6) Deploy On tomcat

—------------------------------------

**1. GIT CLONE and 2. SONARQUBE INTEGRATION**

1. **Installed jenkins in jenkins-ec2 take t2.large**
2. **Created sonarqube-ec2 take t2.large** and installed docker and pull image of sonarqube.

yum -y install docker

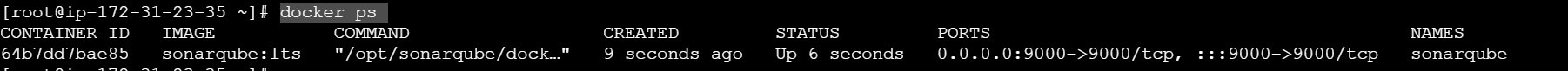
systemctl start docker

docker pull sonarqube (**only download** image.)

docker run -d --name sonarqube -p 9000:9000 sonarqube:lts

(download (if needed) + create + start container.)

Docker ps

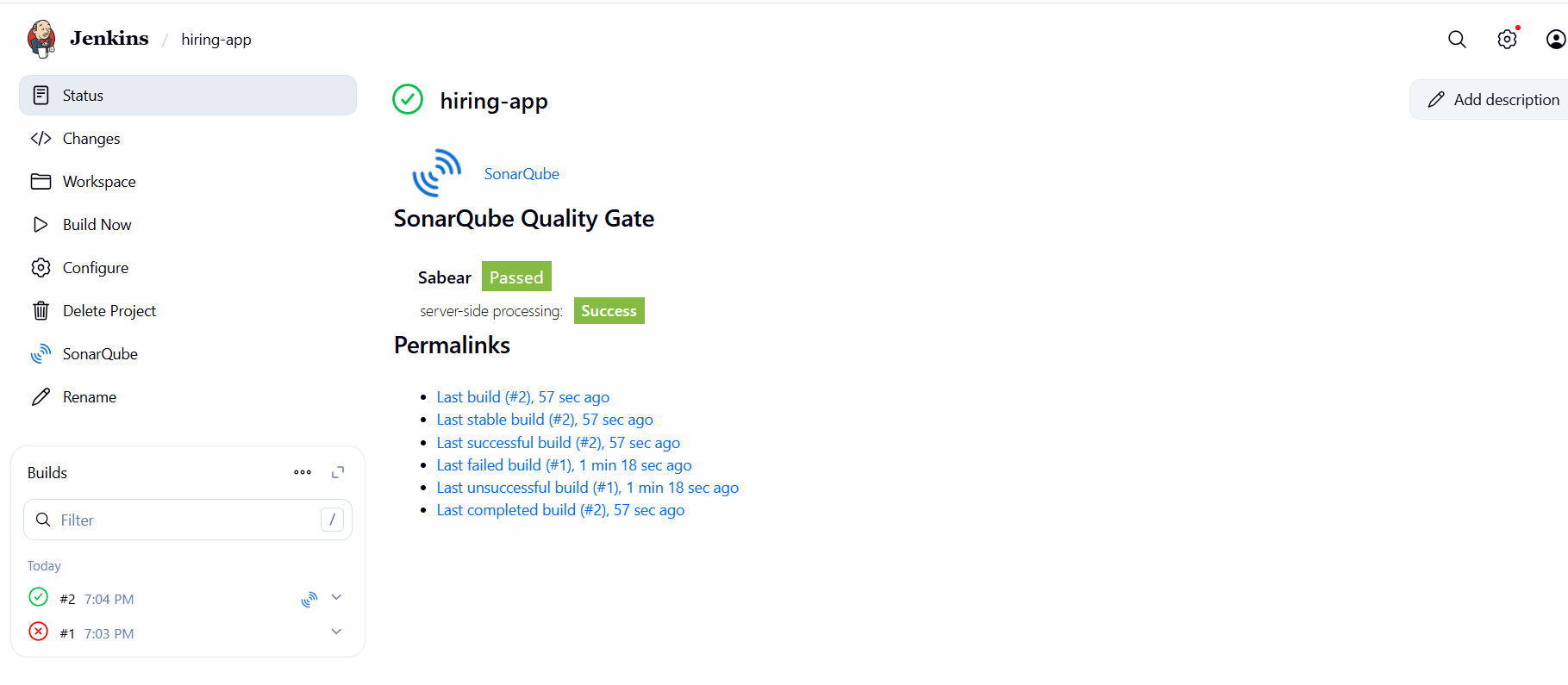
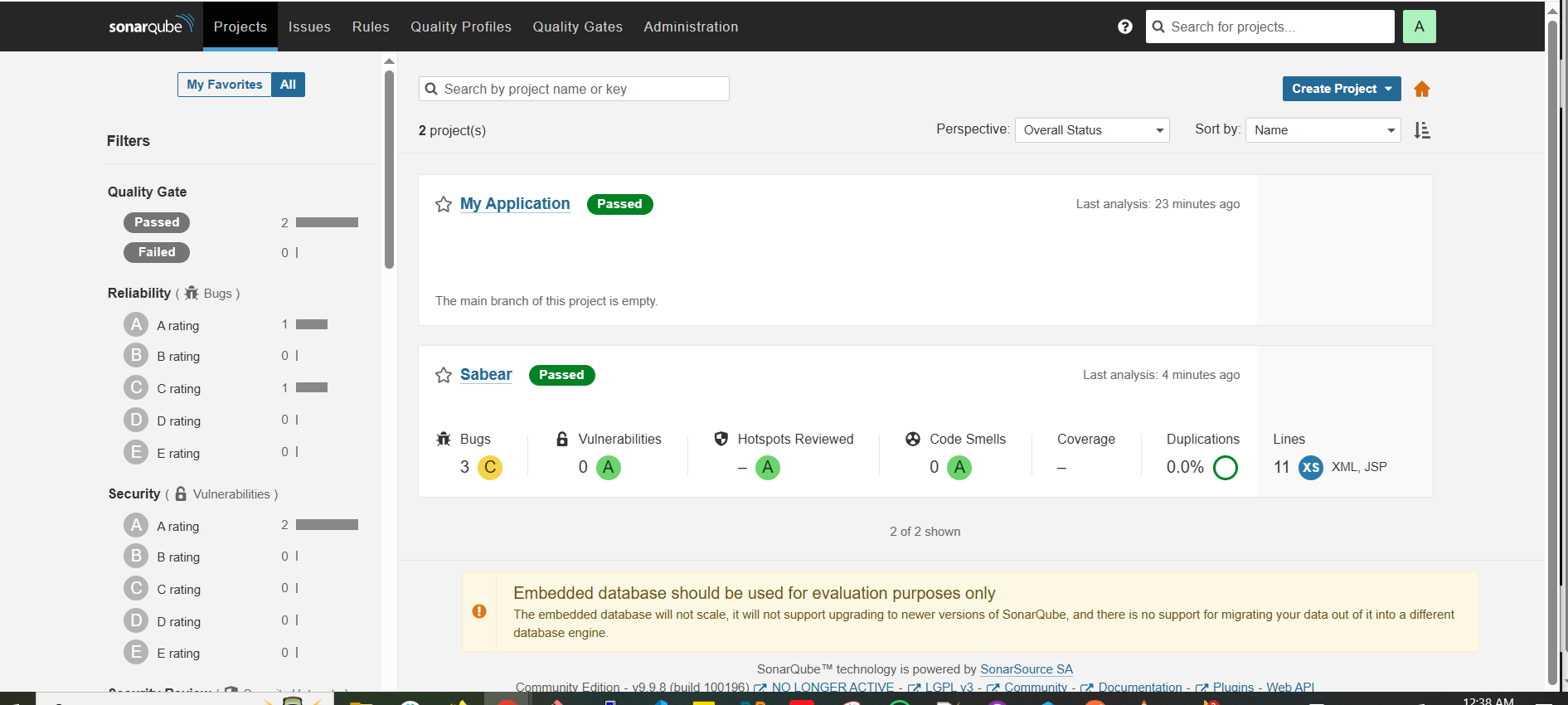
1. 
2. In jenkins master install sonarqube-scanner plugin

Go to **Manage Jenkins → System**.

Scroll to **SonarQube servers** section.

Add:

* **Name** → SonarQube
* **Server URL** → your SonarQube URL (example: http://<sonarqube-server-ip>:9000)
* **Authentication Token** → generate one in SonarQube (My Account → Security → Tokens).

1. Now in jenkins home page, create a job hiring-app and enter the analysis properties
2. <https://github.com/betawins/hiring-app.git>
3. The repo is in main branch same need to configure in job also.
4. 
5. Now start build
6. 
7. 
8. 

**3. MAVEN COMPILATION**

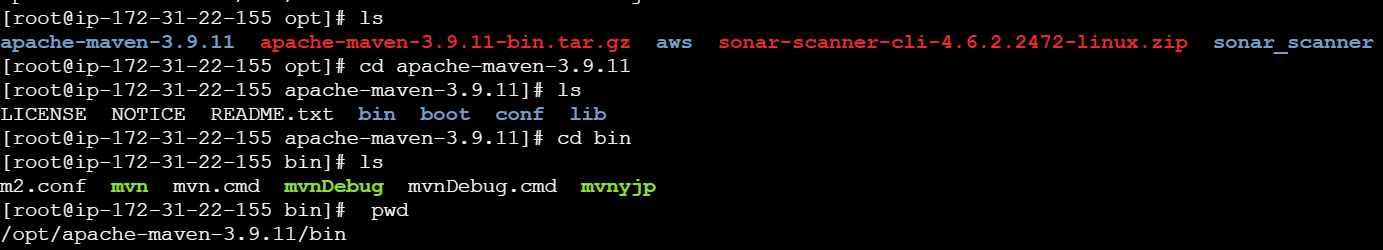
1. download apche maven in jenkins master

cd /opt

wget <https://dlcdn.apache.org/maven/maven-3/3.9.11/binaries/apache-maven-3.9.11-bin.tar.gz>

1. Now extract the tar file

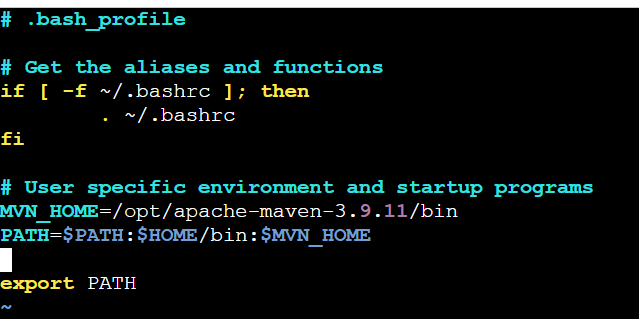
**tar -xvf apache-maven-3.9.11-bin.tar.gz**

1. **cd apache-maven-3.9.11/bin**
2. 
3. **Store global variable in script to access maven –version**
4. **In jenkins server** edit the file and some details → come to root → **vi .bash\_profile**

MVN\_HOME=**/opt/apache-maven-3.9.11/bin**

PATH=$PATH:$HOME/bin:$MVN\_HOME

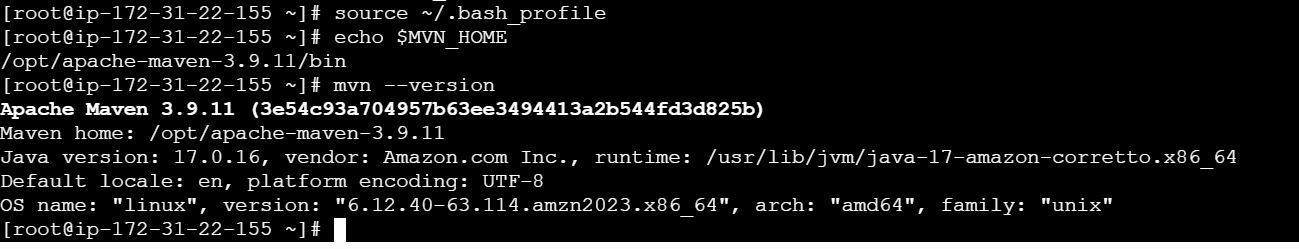
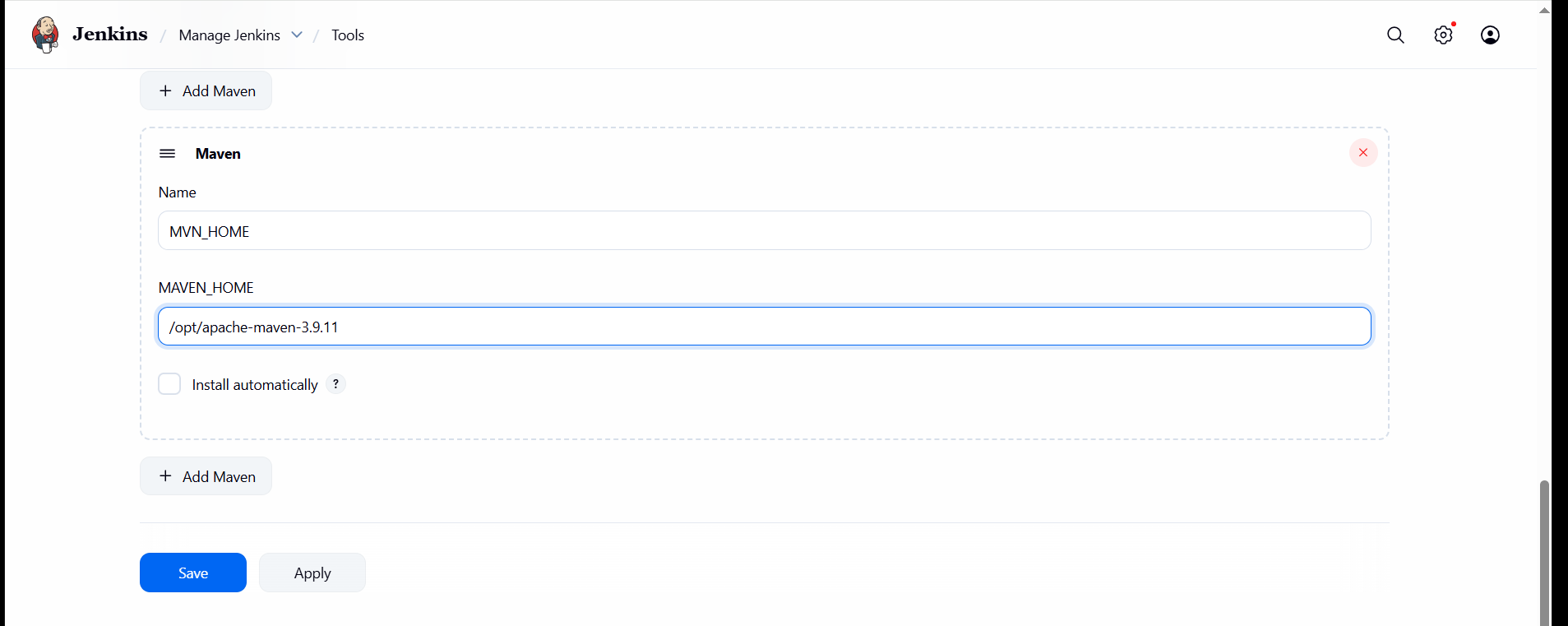
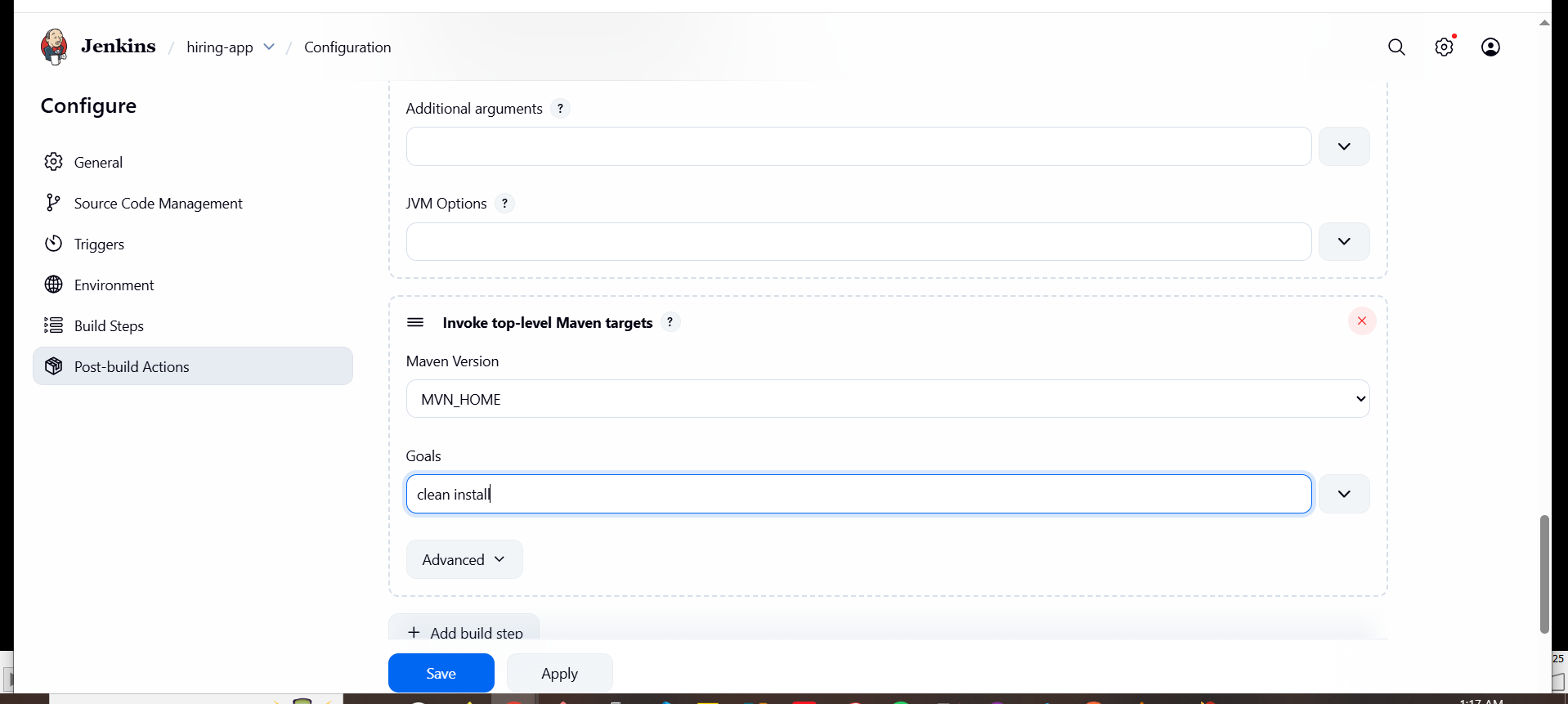
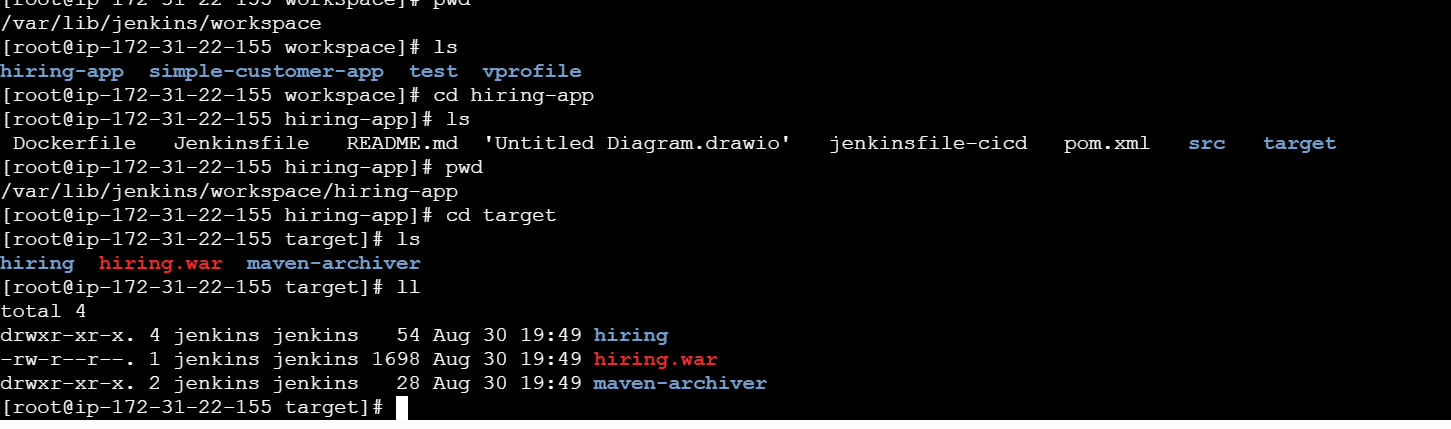
export PATH

1. 
2. Now restart the file and check.

source ~/.bash\_profile

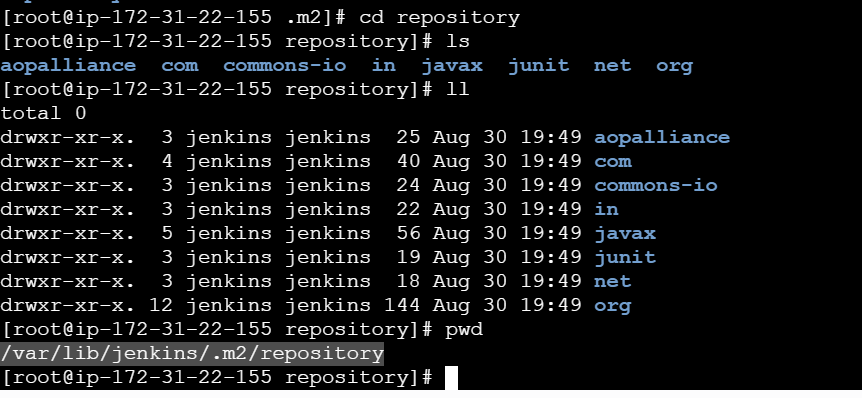
echo $MVN\_HOME

mvn -version

1. 
2. Now goto manage jenkins --> tools --> maven --> /opt/apache-maven-3.9.11
3. 
4. goto job --> add build step --> select invoke top maven -- goals(clean install)
5. Then it will clean and create a package.
6. 
7. Now run the build then git clone done, sonarqube integration done, then maven created a war file.
8. 
9. **/var/lib/jenkins/.m2/repository/in/javahome/hiring/0.1/hiring-0.1.war**
10. **In jenkins server → cd /var/lib/jenkins/workspace/hiring-app/target**
11. ****
12. **(in git repo we have a file pom.xml it connect to maven and download all packages)**
13. We can check dependencies in jenkins server

Cd /var/lib/jenkins/.m2/repository

ll

1. 

**4) Nexus Artifactory**

1. Now create nexus-ec2 with t2.large

install java → **yum install -y java-1.8\***

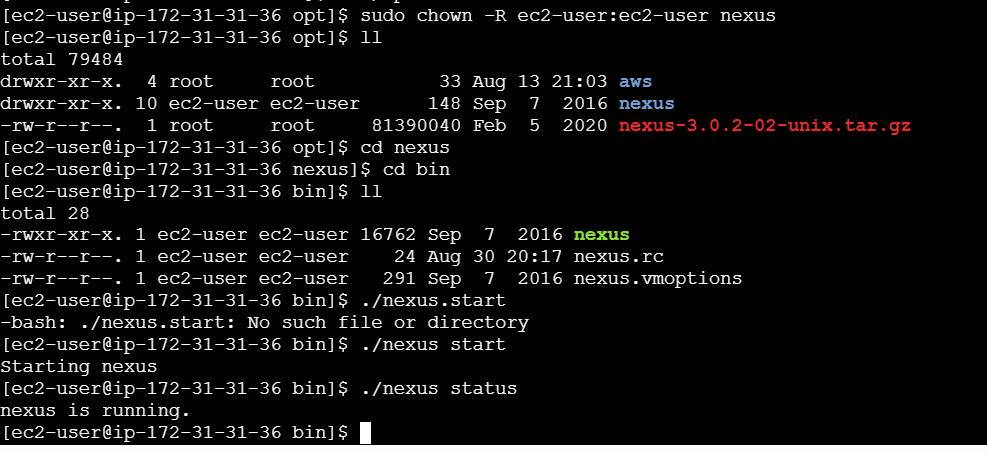
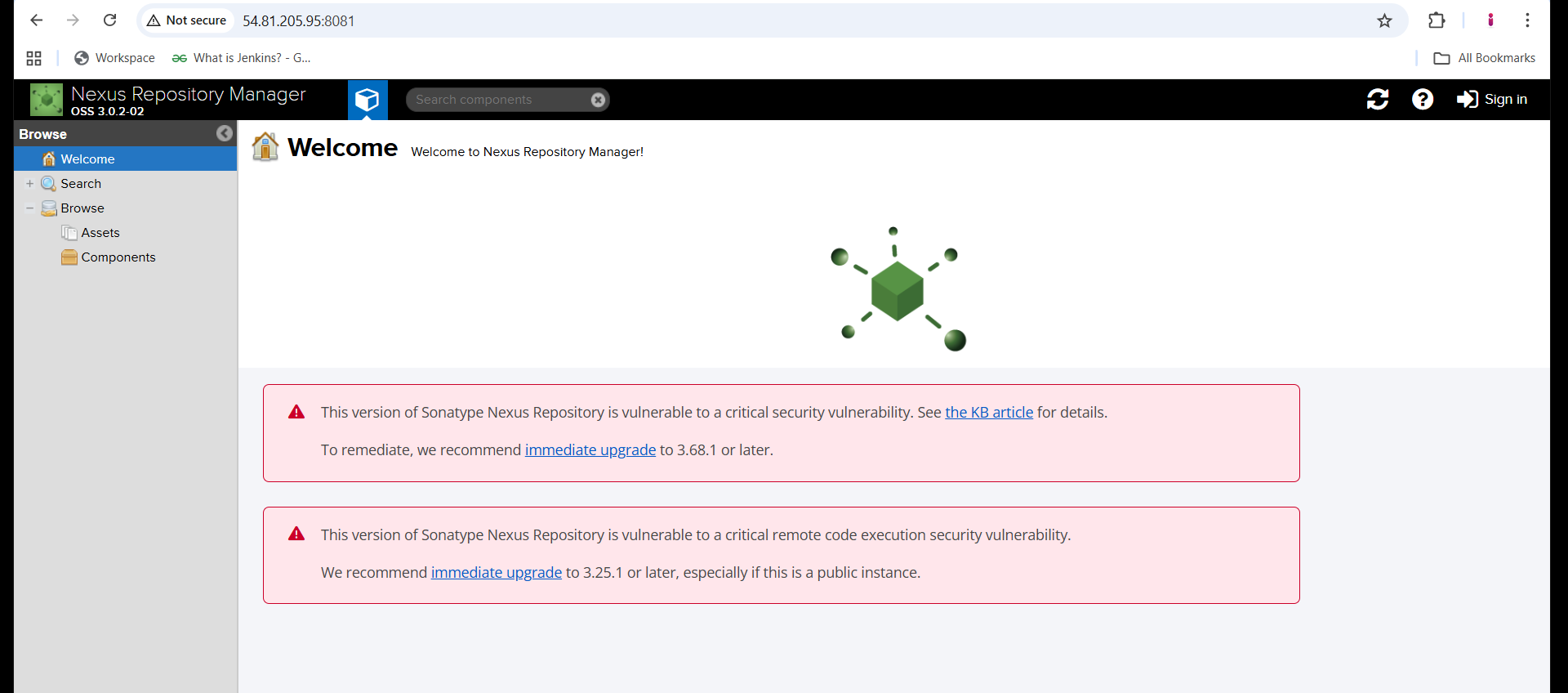
1. And now download nexus

cd /opt

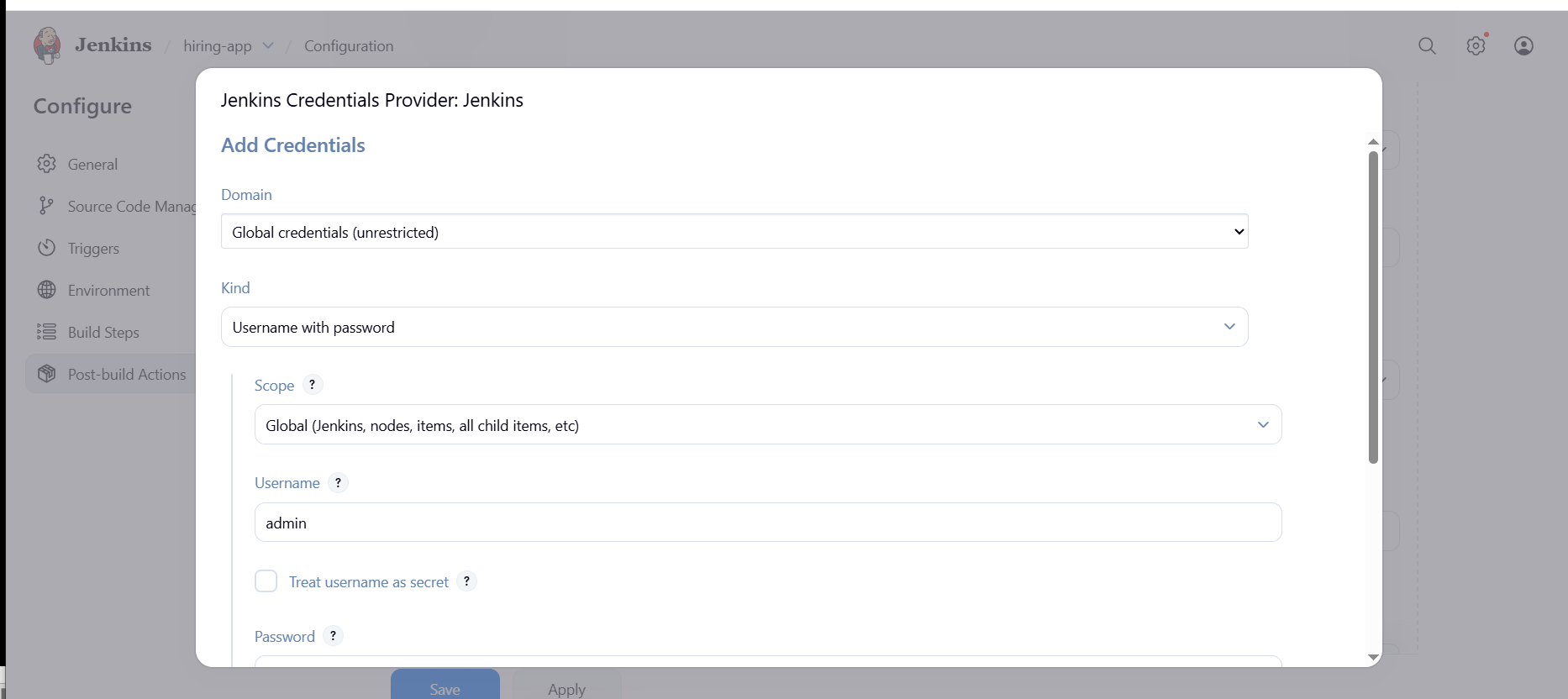
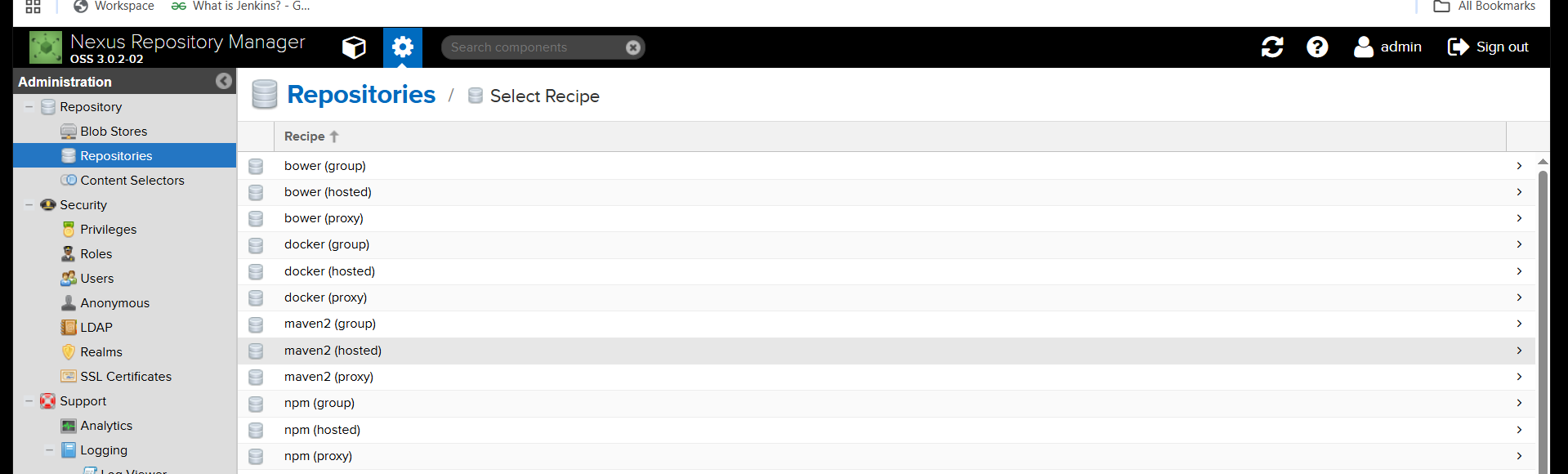
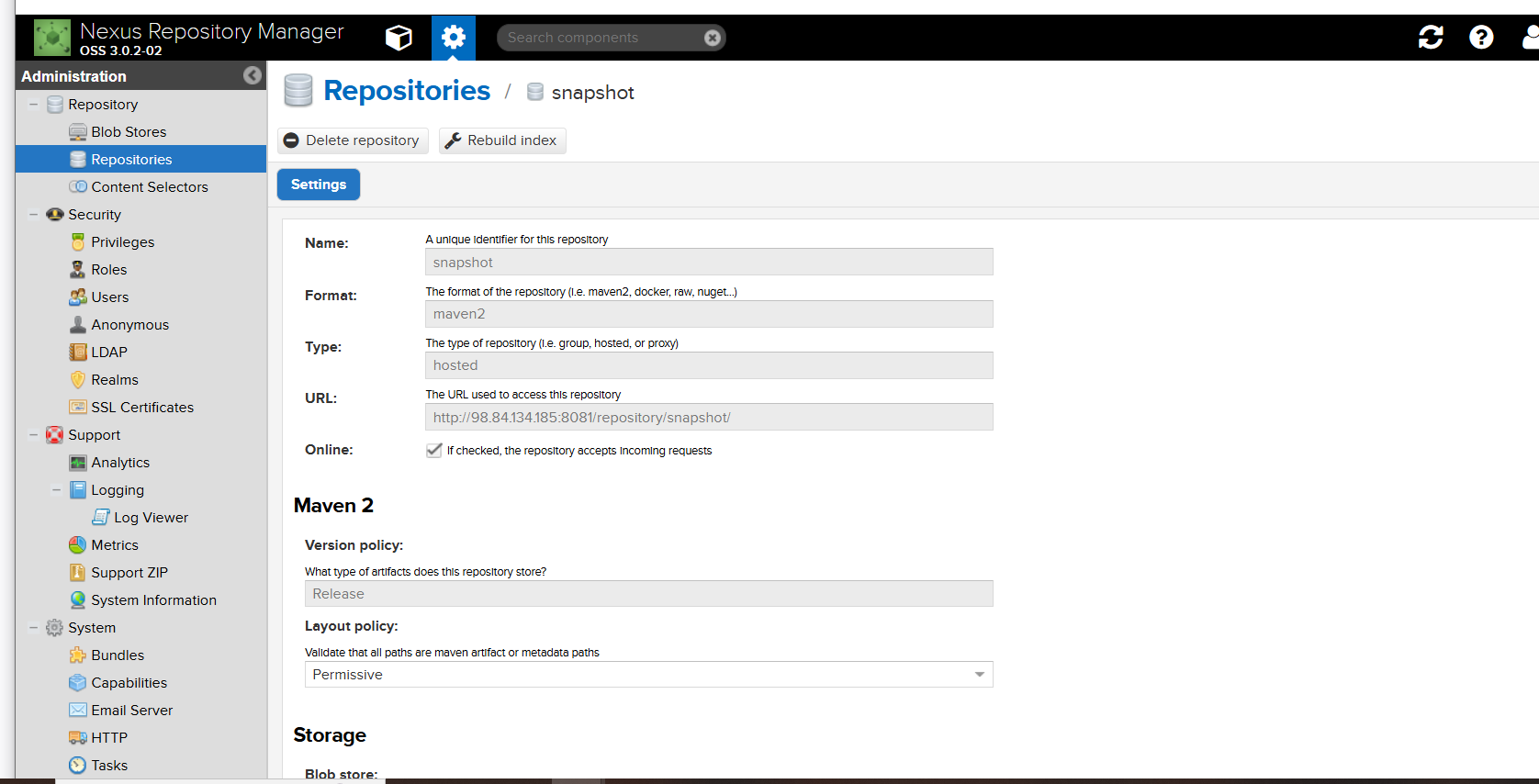
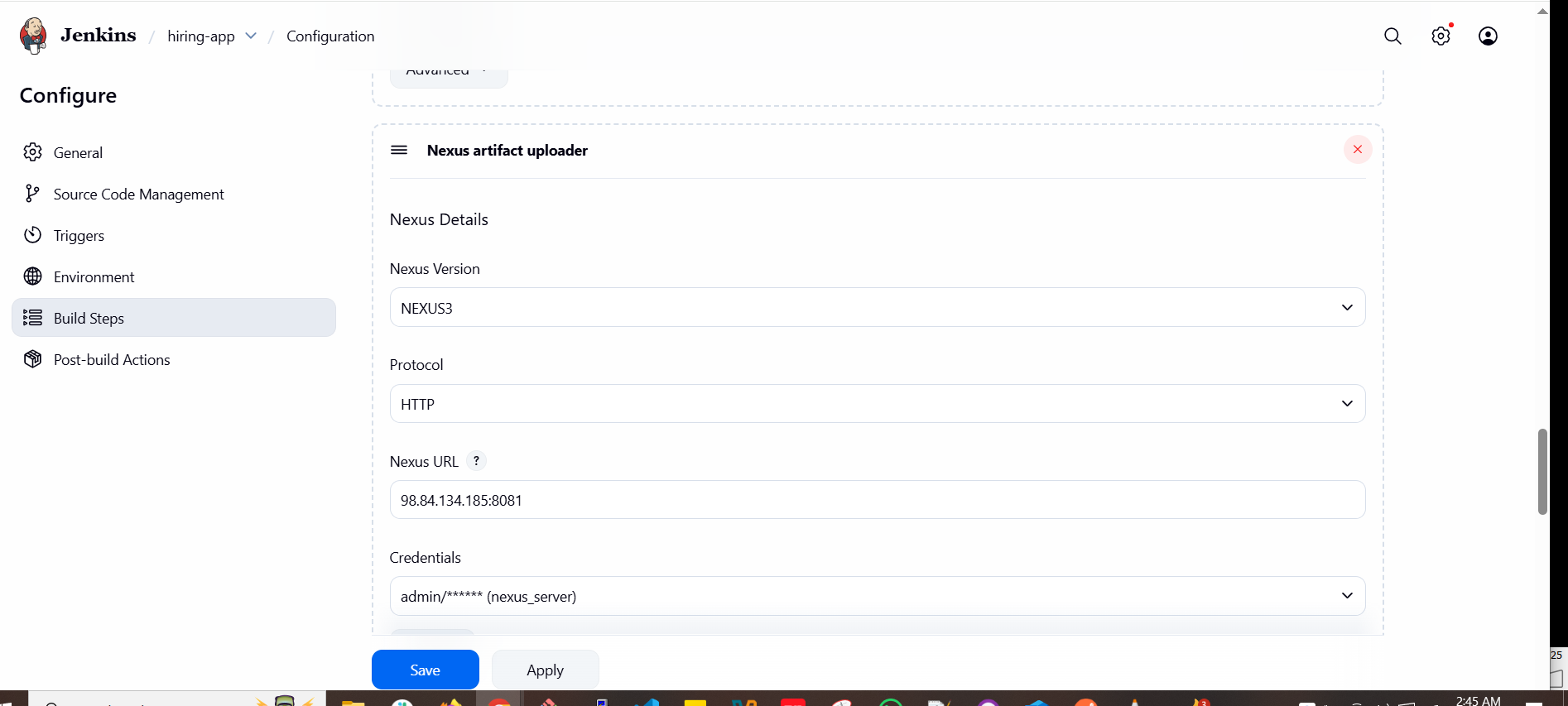
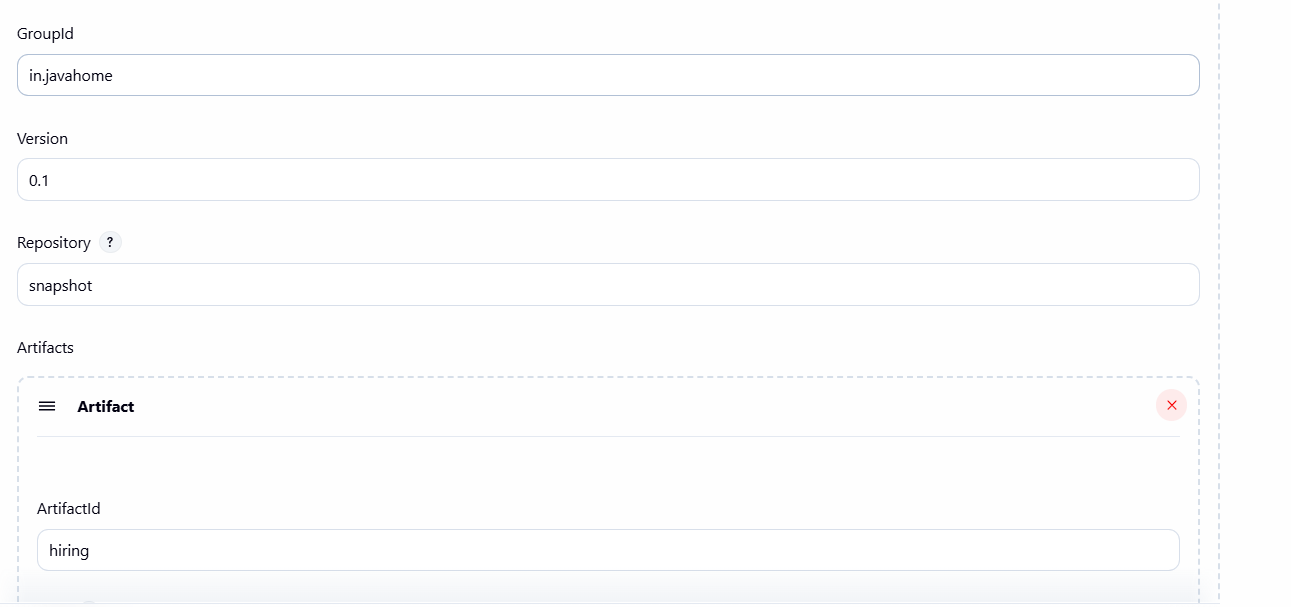
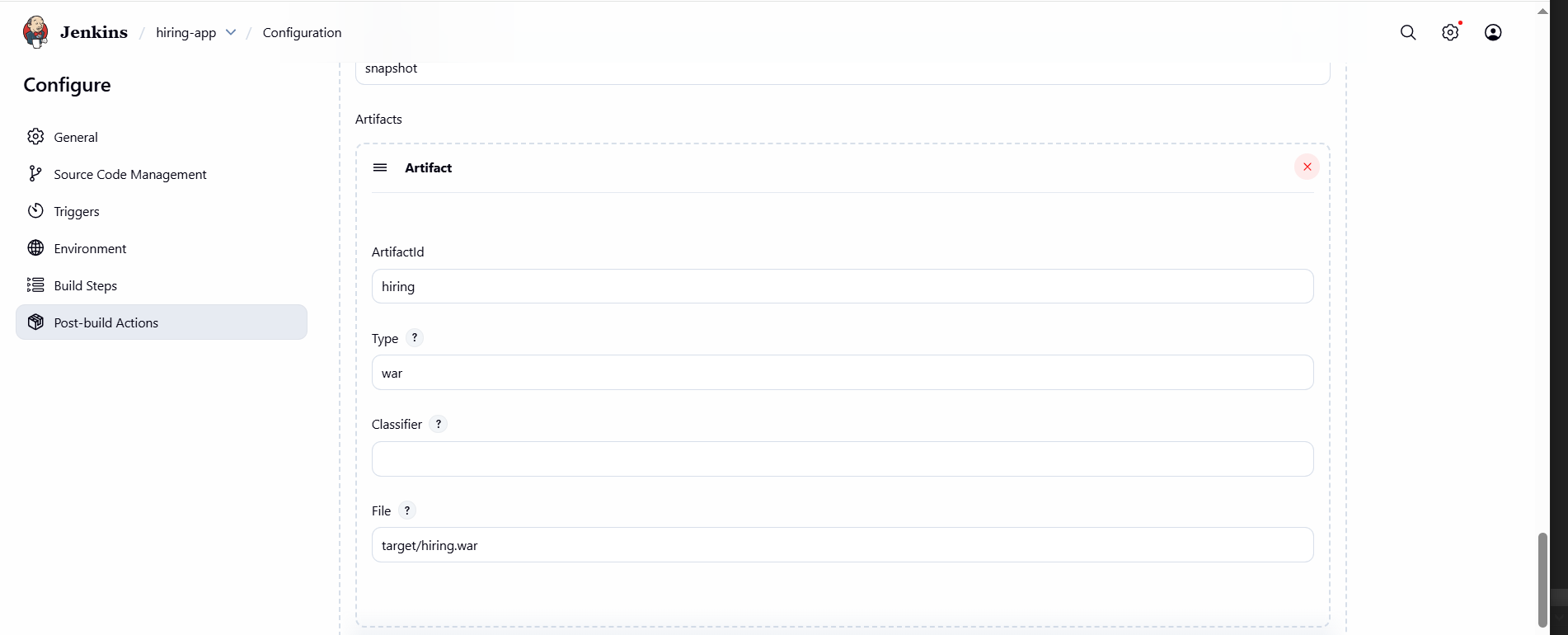
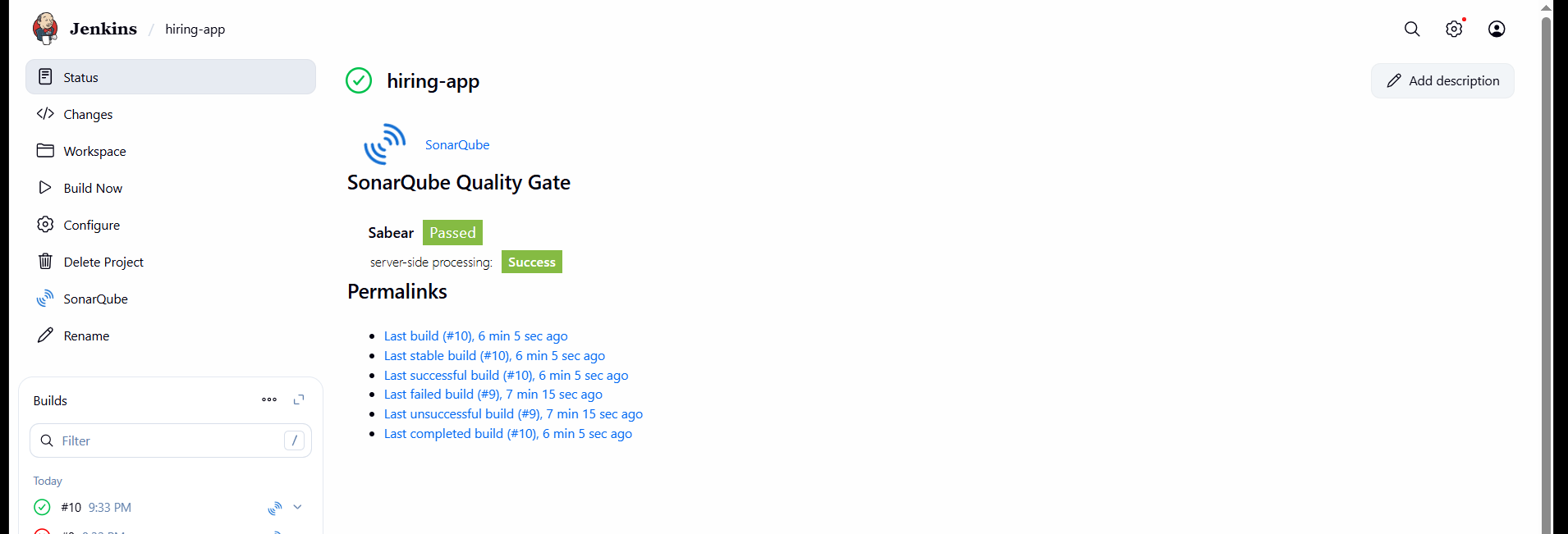
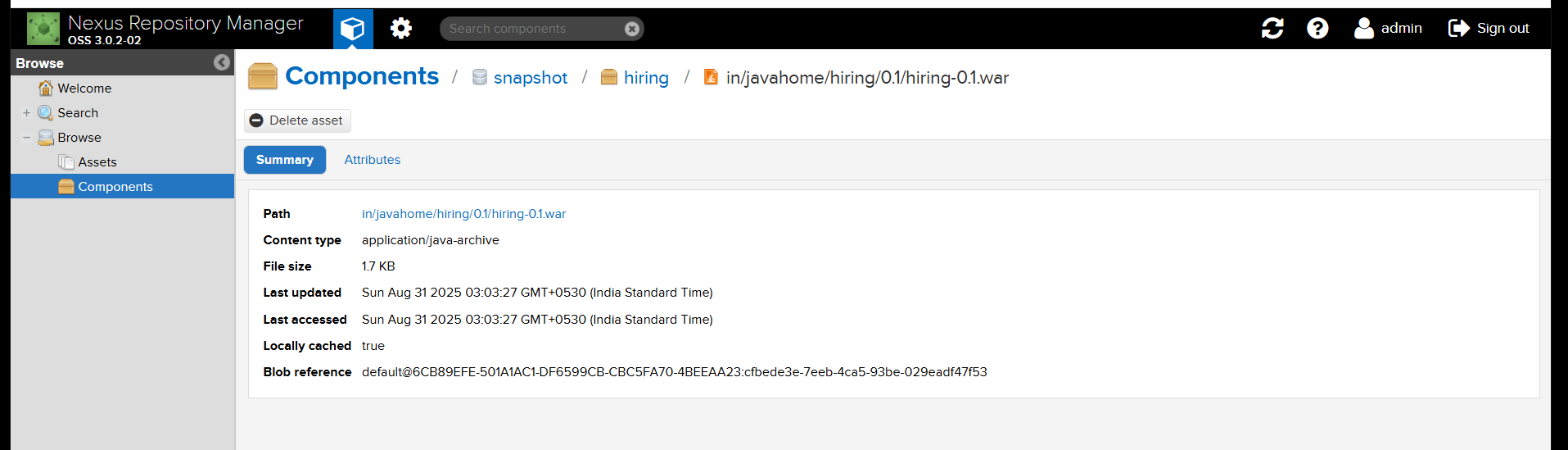
wget <https://sonatype-download.global.ssl.fastly.net/nexus/3/nexus-3.0.2-02-unix.tar.gz>

tar -zxvf nexus-3.0.2-02-unix.tar.gz

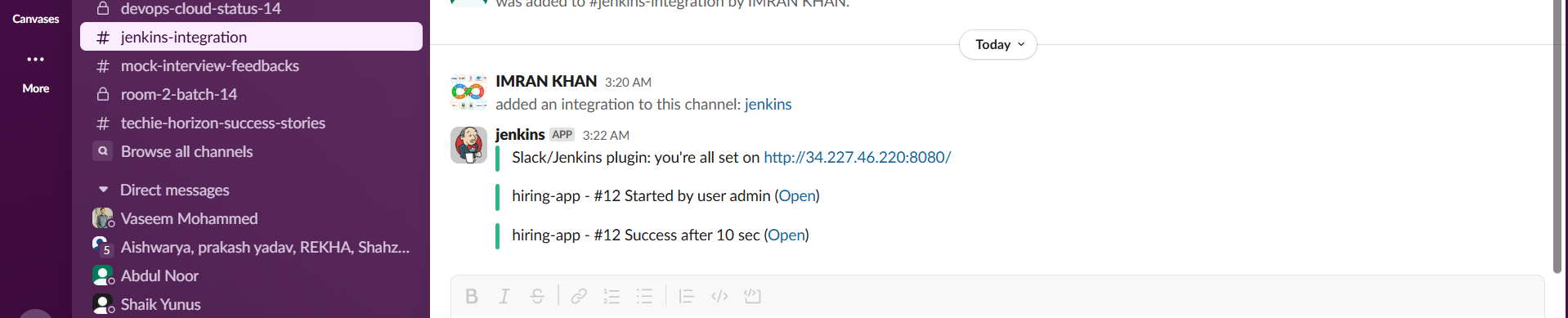
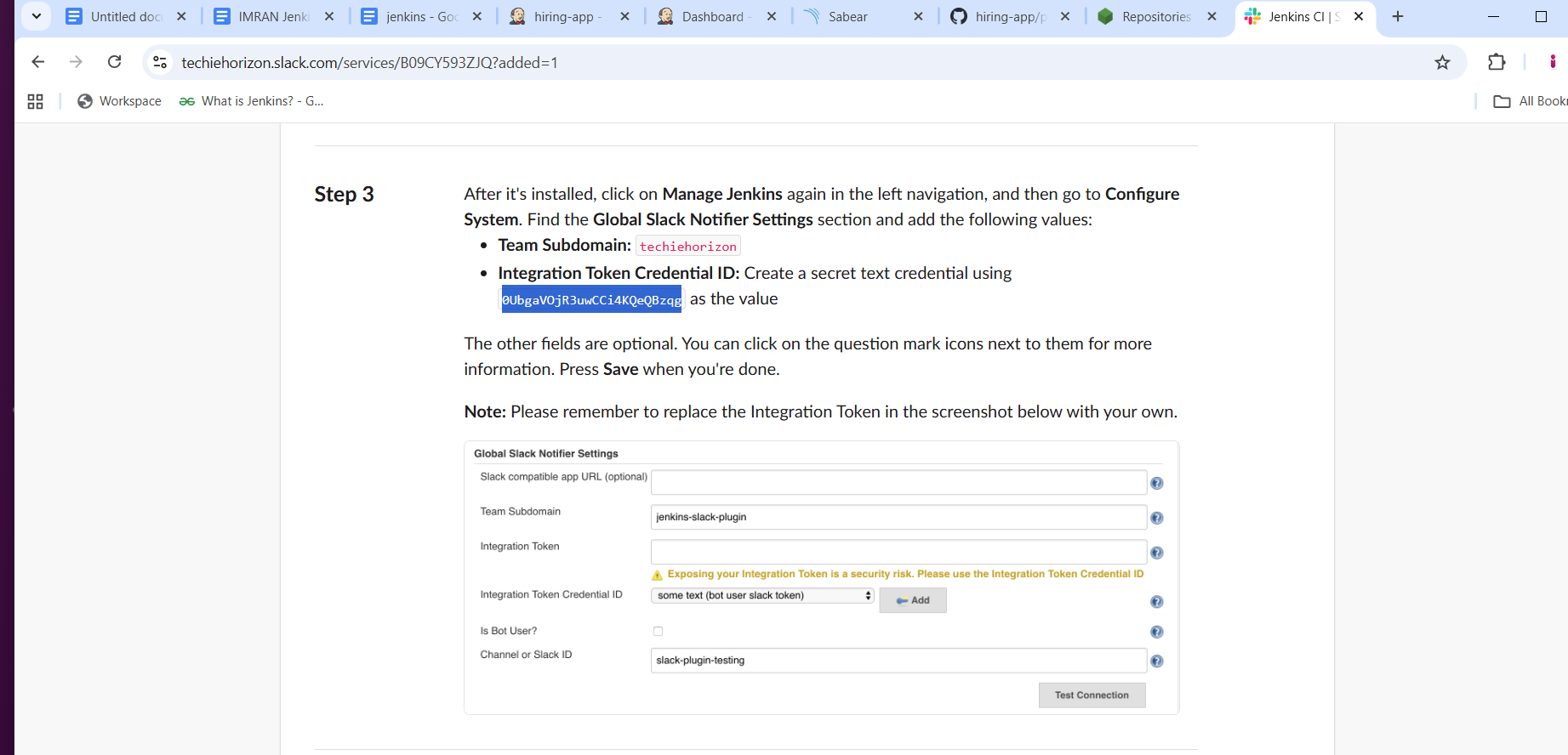
mv /opt/nexus-3.0.2-02 /opt/nexus

1. In nexus ec2 → cd /opt/nexus/bin
2. vi [nexus.rc](http://nexus.rc)
3. Remove # and Add → run\_as\_user="**ec2-user**"
4. Now switch to ec2-user give ownership and start and check status.
5. cd /opt
6. Give permissions → **sudo** **chown -R ec2-user:ec2-user nexus**
7. **cd /opt/nexus/bin**
8. **./nexus start**
9. **./nexus status**
10. 
11. Check in browser with public ip of nexus-ec2 it runs on port number 8081
12. <http://54.81.205.95:8081/>
13. 
14. Now sign in with login details → admin , admin123

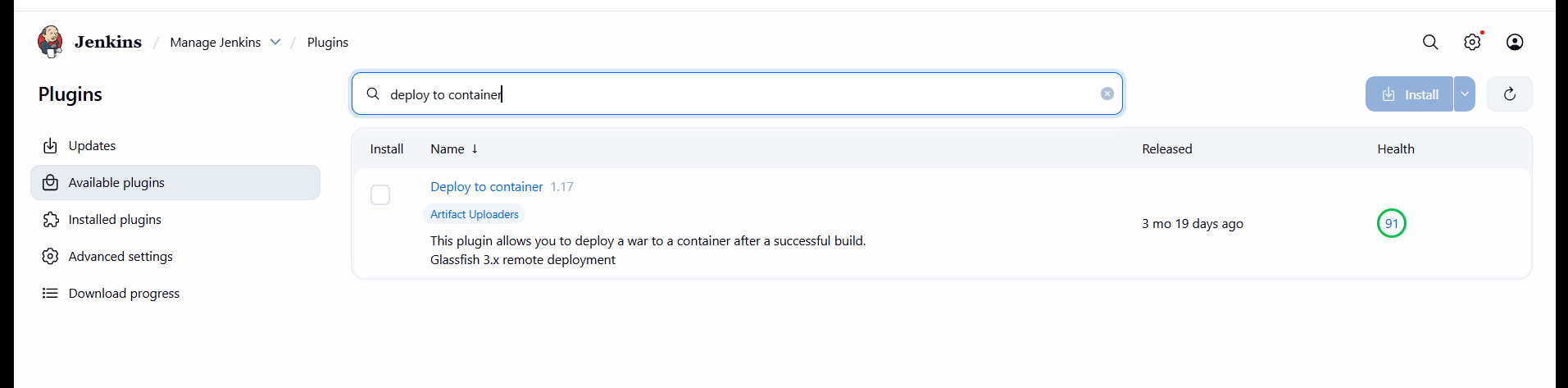
**Integrate jenkins with nexus**

1. **Download plugin “nexus artifact uploader” in jenkins**
2. **Now go to your job → configure → build step → nexus**
3. ****
4. **Add details and credentials also admin, admin123**
5. ****
6. **In git hub repo in pom.xml file we have some info group id and version, bring and enter here and goto nexus create repository → maven2(host) and take release.**
7. ****
8. ****
9. **By checking git repo pom.xml need to fill details give file name → /var/lib/jenkins/workspace/hiring-app/target/hiring.war**
10. **And save try to build.**
11. ****
12. ****
13. ****
14. **Now run build**
15. ****
16. ****
17. **You can check in nexus the war file**
18. ****

**5) Slack Notification**

1. **In jenkins download slack plugin**
2. **In your slack channel see apps add jenkins ci , and add channel and copy the id**
3. ****
4. ****

**6) Deploy On tomcat**

1. **Go to jenkins page Install plugin → deploy to container**
2. ****
3. **Now create tomcat-ec2 and install tomcat.**
4. **yum install -y java-1.8\***

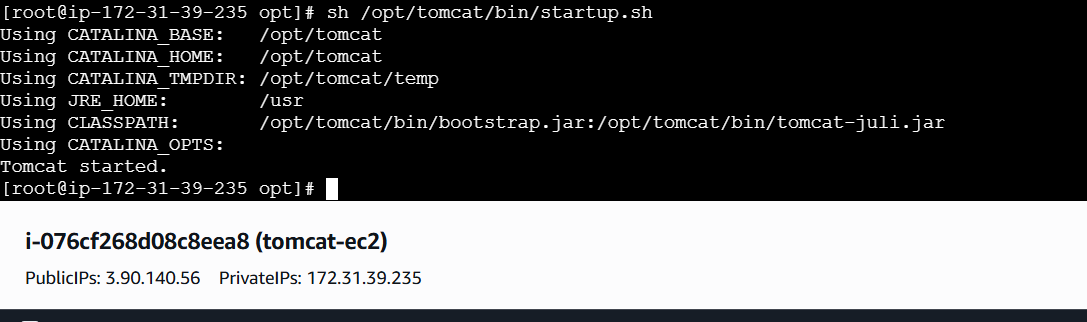
**cd /opt**

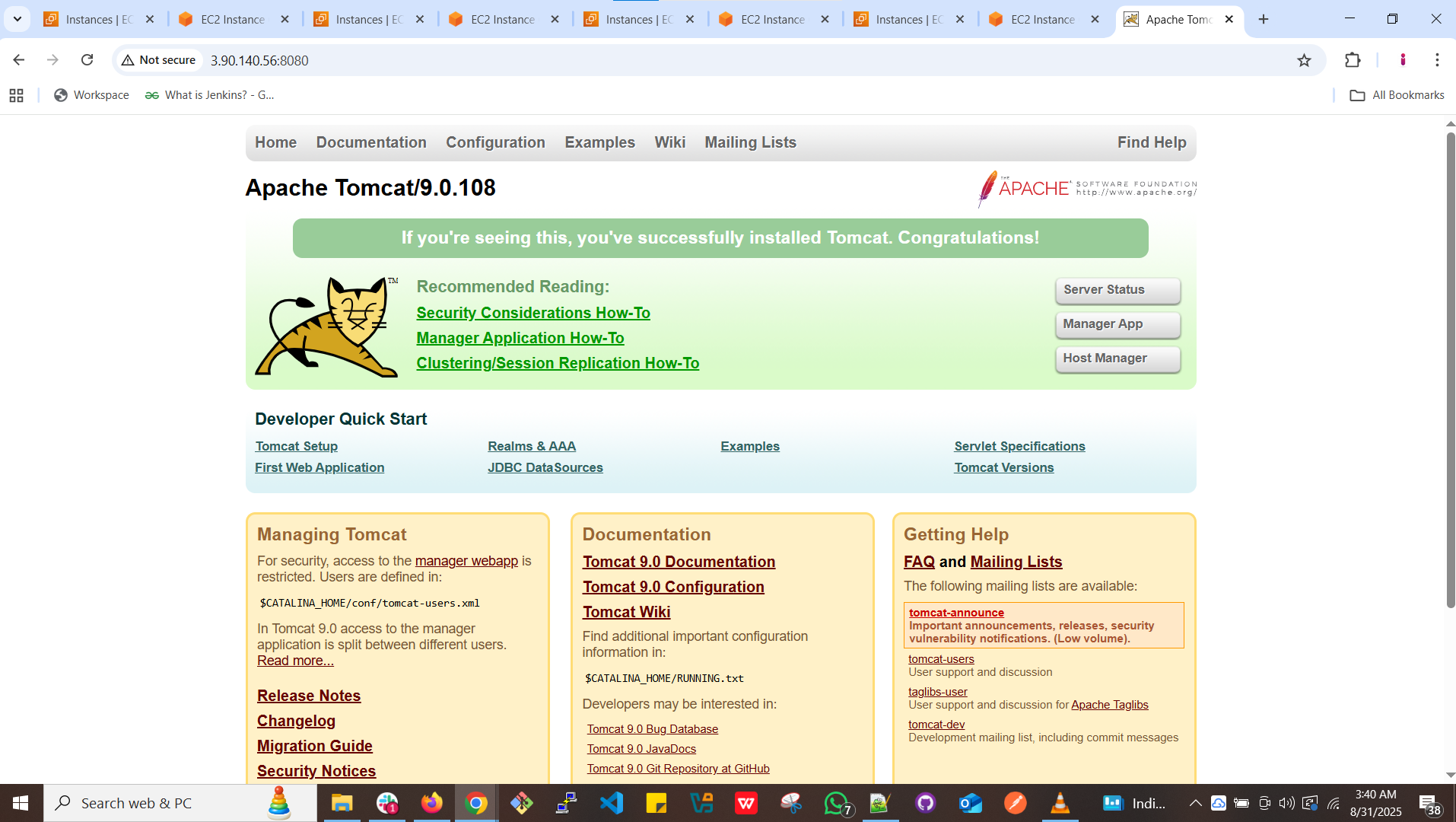
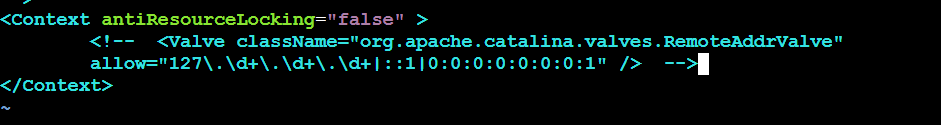
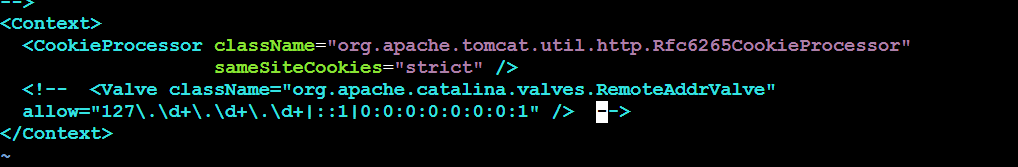
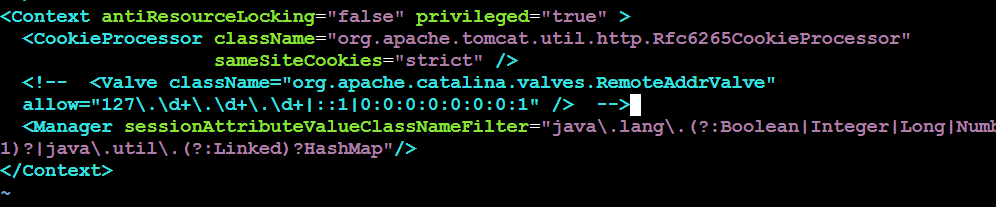
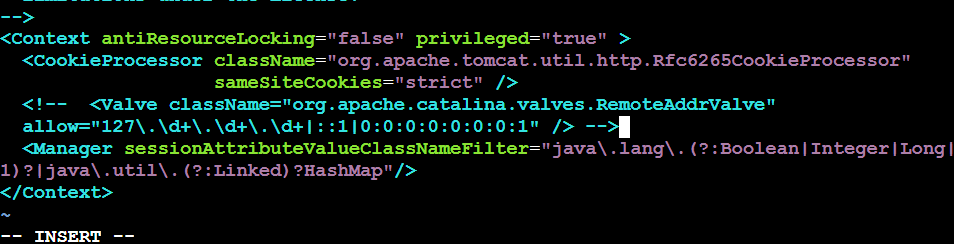
**wget** [**https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz**](https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.108/bin/apache-tomcat-9.0.108.tar.gz)

**tar xvf** [**apache-tomcat-9.0.108.tar.gz**](http://apache-tomcat-9.0.108.tar.gz)

**sudo mv apache-tomcat-9.0.108 /opt/tomcat**

**sh /opt/tomcat/bin/**[**startup.sh**](http://startup.sh)

****

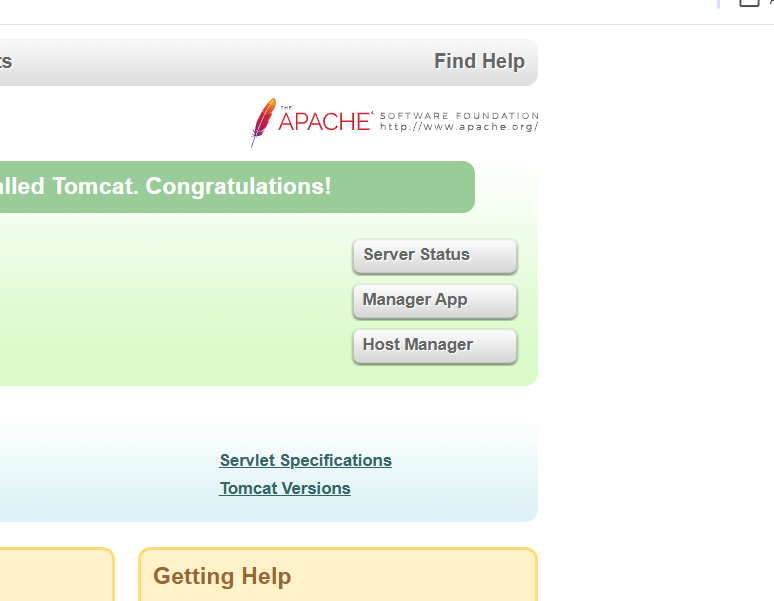
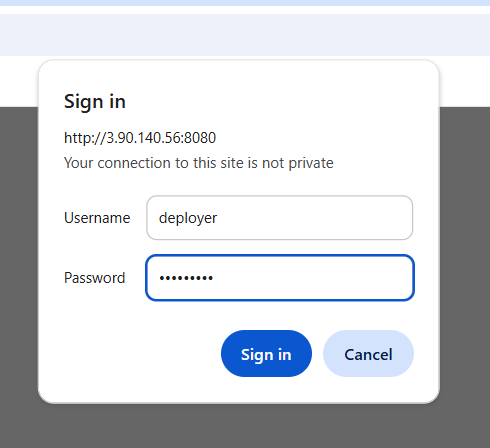
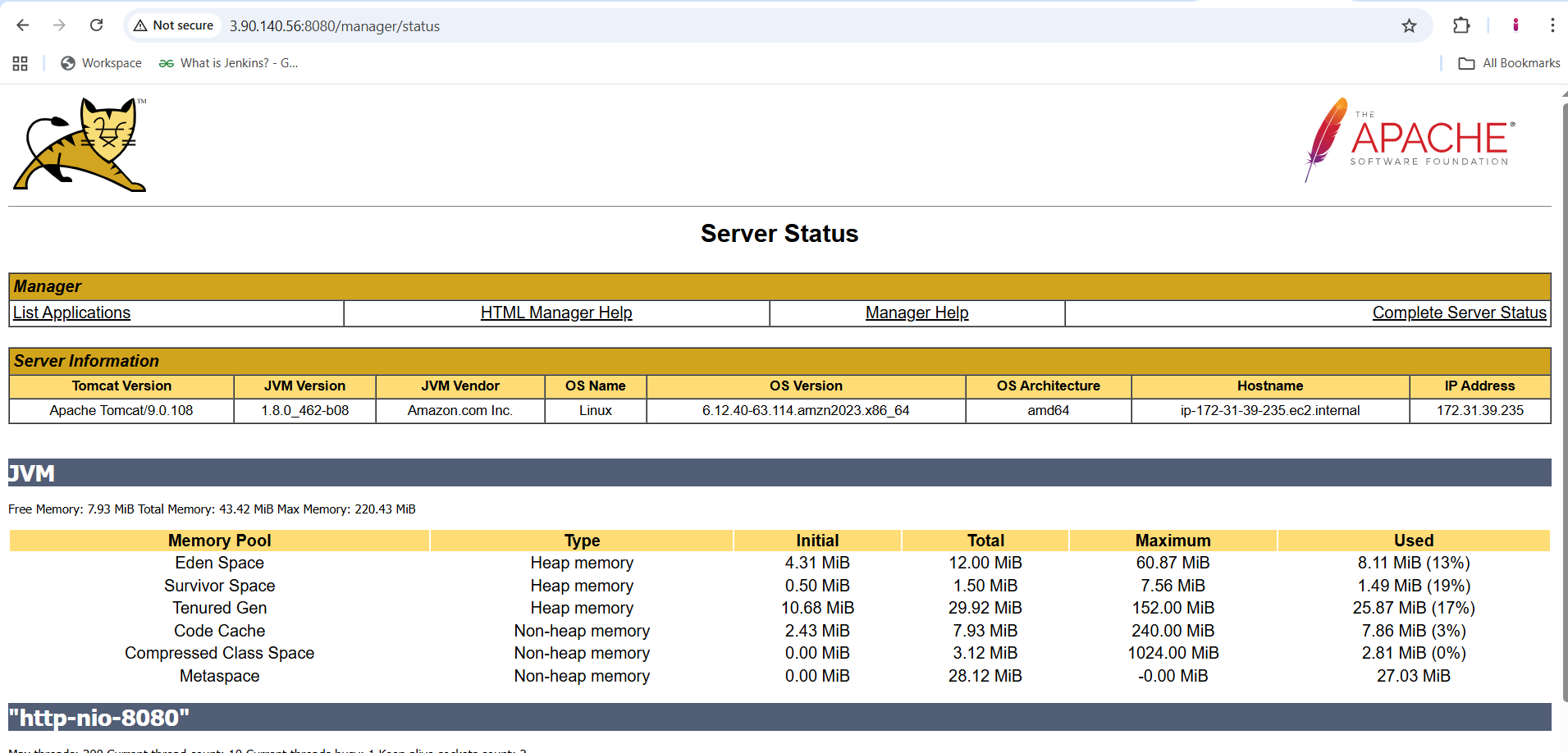
1. ****
2. **Now create user in tomcat.**
3. **find / -name context.xml**
4. **Take file webapps and edit for uncomment the four files**
5. **vi /opt/tomcat/webapps/docs/META-INF/context.xml**
6. ****
7. **vi /opt/tomcat/webapps/examples/META-INF/context.xml**
8. ****
9. **vi /opt/tomcat/webapps/host-manager/META-INF/context.xml**
10. ****
11. **vi /opt/tomcat/webapps/manager/META-INF/context.xml**
12. ****
13. **Now goto conf in xml file add the below user details as shown in box.**

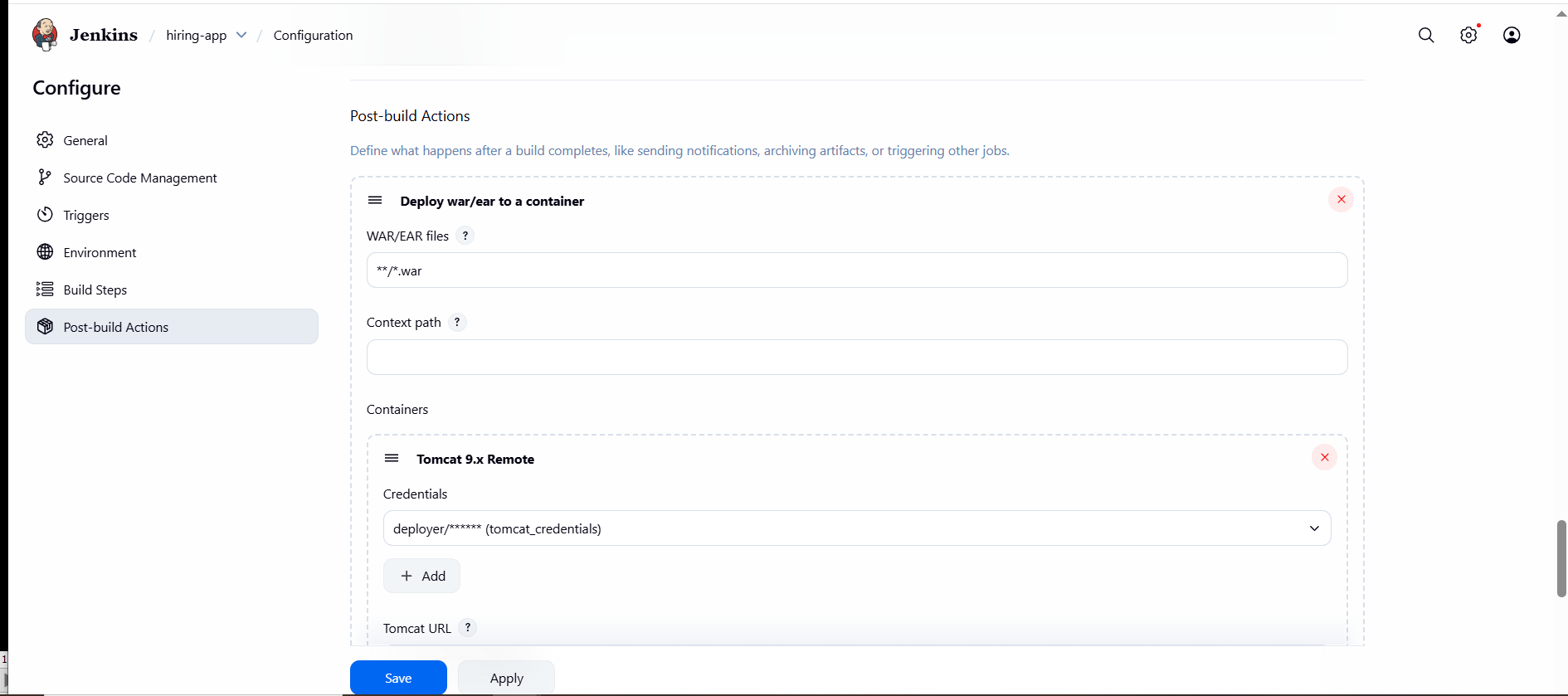
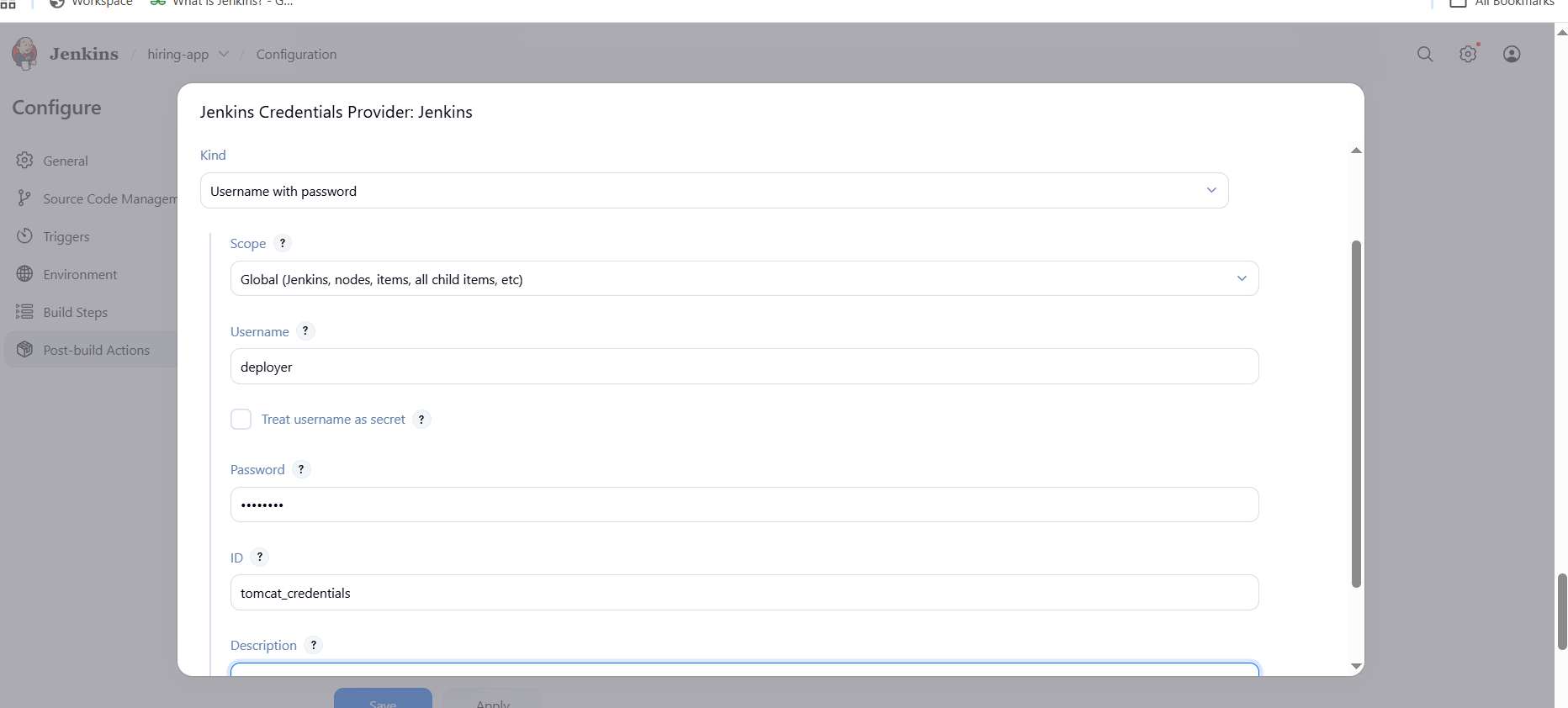
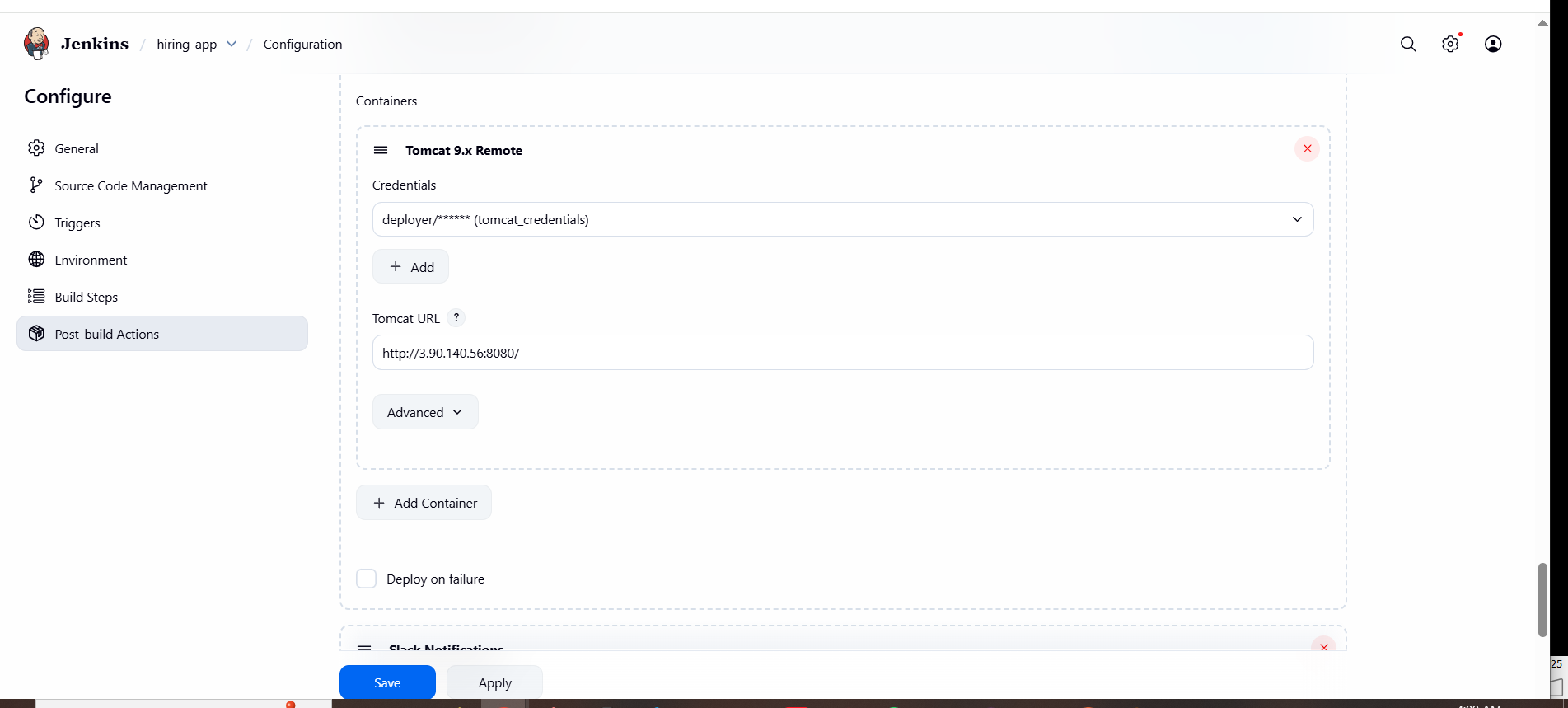
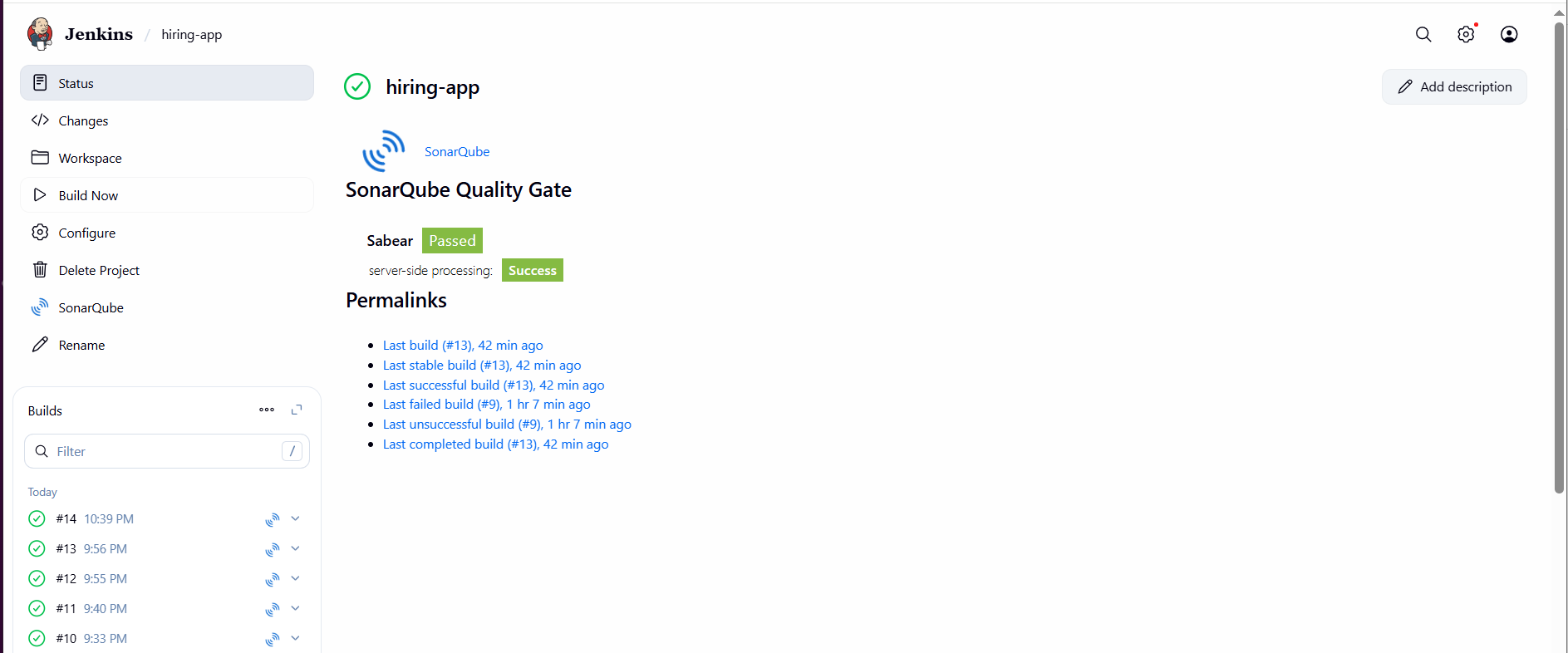
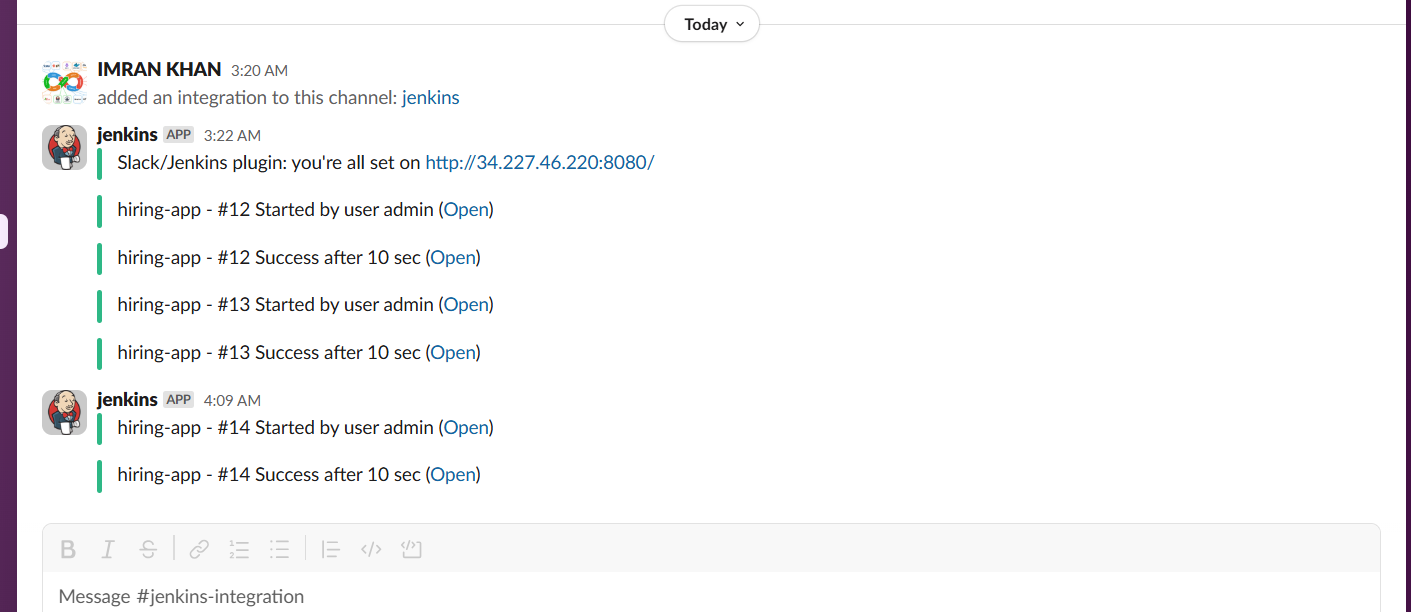
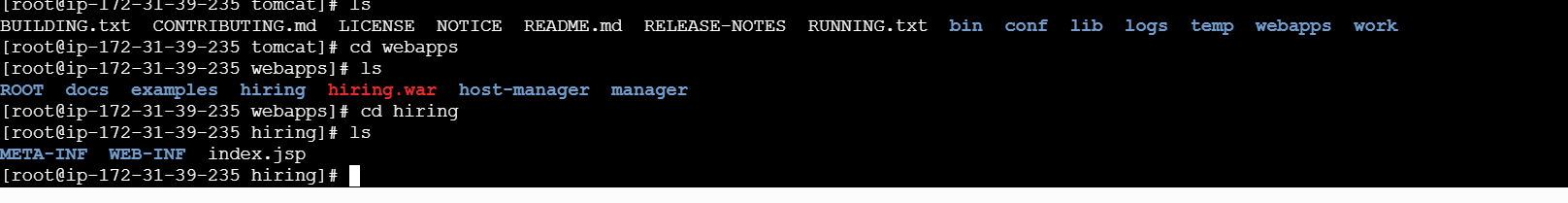
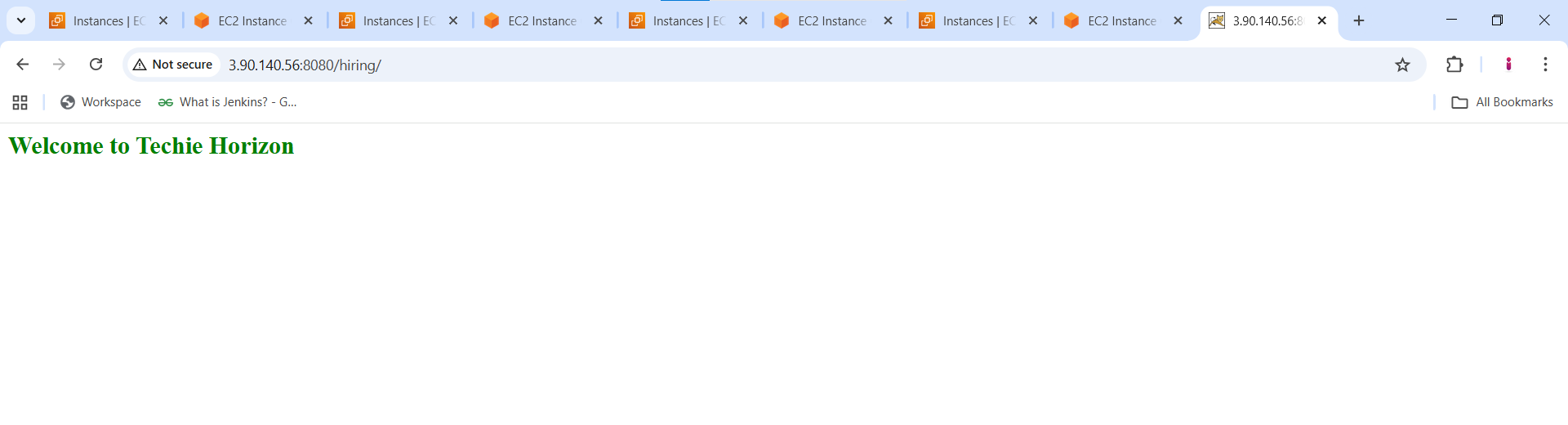
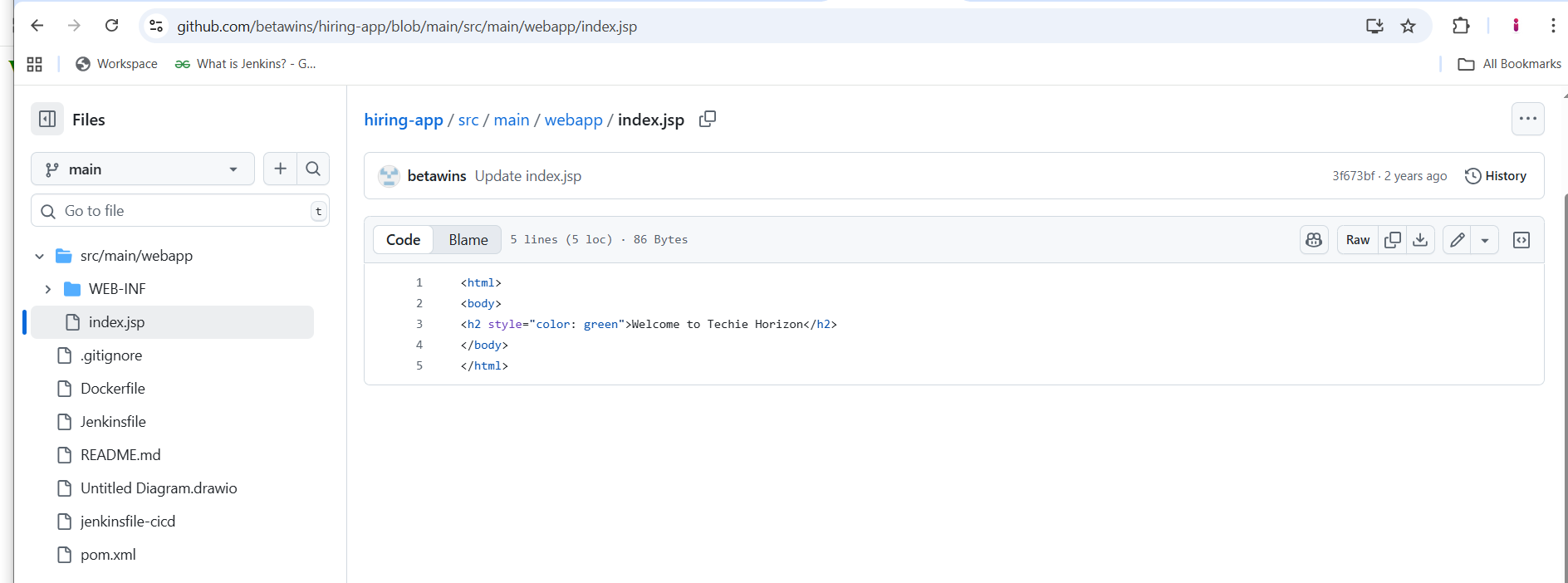
**cd /opt/tomcat/conf**

**vi tomcat-users.xml**

****

****

1. **sh /opt/tomcat/bin/shutdown.sh**
2. **sh /opt/tomcat/bin/startup.sh**
3. **Now in web page click on server status**
4. ****
5. **Enter user deployer , password deployer**
6. ****
7. ****
8. **Just go to jenkins add these credentials deployer and take WAR/EAR files as \*\*/\*.war**

1. ****
2. ****
3. ****
4. ****
5. ****
6. **Now our deployment successfully done**
7. **Now you can check in tomcat cli webapps one file is shown**
8. **Cd /opt/tomcat/webapps/hiring**
9. ****
10. **And you can check in tomcat at browser we got our application is live.**
11. **http://3.90.140.56:8080/hiring/**
12. ****
13. **And you can confirm the content by checking git repo code.**
14. ****

**2) Setup a jenkins CICD pipeline using Declarative pipeline using feature-1.1 branch.**

<https://github.com/betawins/sabear_simplecutomerapp/tree/feature-1.1>

stages:

1) Git Clone

2) Sonarqube Integration

3) Maven Compilation

4) Nexus Artifactory

5) Slack Notification

6) Deploy On tomcat

—----------------------------------------------------------------------------------------------------------------------

Pre Requisite done.

**# 1. Clone the original repo directly**

**git clone https://github.com/betawins/sabear\_simplecutomerapp.git**

**cd sabear\_simplecutomerapp**

**# 2. Connect to your GitHub repo**

**Go to GitHub → Create a new empty repository (without README).  
 Example: https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git**

**Add it as remote**

**git remote add origin https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git**

**# 3. Push all branches from original repo**

Option 1

git checkout feature-1.1

git push origin feature-1.1

Option 2

git push origin 'refs/remotes/origin/\*:refs/heads/\*'

—-------------------------------------------------------- OR ---------------------------------------

1. Fork the required git hub repo from another github account.

# 1. Clone your fork (your GitHub repo)

git clone https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git

cd sabear\_simplecutomerapp

# 2. Add the original repo as another remote

git remote add upstream https://github.com/betawins/sabear\_simplecutomerapp.git

# 3. Fetch all branches from original repo

git fetch upstream

# 4. Checkout the feature branch you want

git checkout -b feature-1.1 upstream/feature-1.1

# 5. Push it to your repo (fork)

git push origin feature-1.1

—-----------------------------------------------------------------------------------------------------

# **Jenkins CI/CD Pipeline for Simple Customer App**

## **1. Job Overview**

**The Simple Customer App Jenkins job automates the following tasks:**

1. **Checkout code from GitHub repository**
2. **Build the project using Maven**
3. **Run SonarQube code analysis**
4. **Deploy the WAR file to Nexus repository**
5. **Deploy the WAR file to Tomcat server**
6. **Send Slack notifications for build status**

## **2. Prerequisites**

**Before creating the Jenkins job, ensure the following are set up:**

1. **Jenkins server with required plugins:**
   1. **Git Plugin**
   2. **Maven Integration Plugin**
   3. **SonarQube Scanner Plugin**
   4. **Slack Notification Plugin**
2. **Maven installed on Jenkins (e.g., Maven 3.8.4)**
3. **SonarQube server configured in Jenkins**
4. **Nexus repository with admin credentials**
5. **Tomcat server installed and running**
6. **Jenkins credentials created for:**
   1. **Nexus (nexus-creds)**
   2. **Tomcat (tomcat-credentials)**
   3. **Slack (slack\_notification)**

## **3. GitHub Repository**

* **Repository URL:  
   https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git**
* **Branch: feature-1.1**

**The repository contains the SimpleCustomerApp project, structured as a Maven WAR project.**

**4. Jenkinsfile Configuration**

**pipeline {**

**agent any**

**environment {**

**NEXUS\_CRED = 'nexus-creds'**

**TOMCAT\_CRED = 'tomcat-credentials'**

**}**

**stages {**

**stage('Checkout SCM') {**

**steps {**

**git url: 'https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git', branch: 'feature-1.1'**

**}**

**}**

**stage('Build') {**

**steps {**

**tool name: 'Maven-3.8.4', type: 'maven'**

**sh 'mvn clean package -DskipTests'**

**}**

**}**

**stage('SonarQube Analysis') {**

**steps {**

**withSonarQubeEnv('SonarQube') {**

**sh 'mvn sonar:sonar'**

**}**

**}**

**}**

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

**steps {**

**withCredentials([usernamePassword(credentialsId: "${NEXUS\_CRED}", usernameVariable: 'NEXUS\_USER', passwordVariable: 'NEXUS\_PASS')]) {**

**sh '''**

**mvn deploy -DskipTests \**

**-Dnexus.username=$NEXUS\_USER \**

**-Dnexus.password=$NEXUS\_PASS \**

**--settings /var/lib/jenkins/.m2/settings.xml**

**'''**

**}**

**}**

**}**

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

**steps {**

**withCredentials([usernamePassword(credentialsId: "${TOMCAT\_CRED}", usernameVariable: 'TOMCAT\_USER', passwordVariable: 'TOMCAT\_PASS')]) {**

**sh '''**

**WAR\_FILE=$(ls target/\*.war | head -n 1)**

**WAR\_NAME=$(basename $WAR\_FILE .war | tr '[:upper:]' '[:lower:]')**

**echo "Deploying $WAR\_FILE to Tomcat at context path /$WAR\_NAME..."**

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

**-T $WAR\_FILE \**

**"http://54.145.142.96:8080/manager/text/deploy?path=/$WAR\_NAME&update=true"**

**'''**

**}**

**}**

**}**

**stage('Slack Notification') {**

**steps {**

**slackSend(**

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

**color: 'good',**

**message: "Hi Team, Jenkins pipeline for \*Simple Customer App\* has finished successfully! ✅\nDeployed by: Imran Khan"**

**)**

**}**

**}**

**}**

**post {**

**always {**

**echo 'Pipeline finished'**

**}**

**failure {**

**slackSend(**

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

**color: 'danger',**

**message: "⚠️ Jenkins pipeline for \*Simple Customer App\* failed! Please check."**

**)**

**}**

**}**

**}**

## **5. Pipeline Execution Steps**

### **Step 1: Checkout SCM**

* **Jenkins fetches the code from GitHub branch feature-1.1.**
* **Command executed:**

**git clone -b feature-1.1 https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git**

### **Step 2: Build with Maven**

* **Project is cleaned and packaged as a WAR file.**
* **Skip tests for faster builds.**
* **Output WAR: target/SimpleCustomerApp-1.0-SNAPSHOT.war**

### **Step 3: SonarQube Analysis**

* **Static code analysis is performed.**
* **Report uploaded to SonarQube dashboard.**

### **Step 4: Deploy to Nexus**

* **WAR is deployed to Nexus snapshot repository.**
* **Nexus credentials are used securely.**

### **Step 5: Deploy to Tomcat**

* **WAR file is deployed to Tomcat Manager at path /simplecustomerapp.**
* **Command executed:**

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

**-T target/SimpleCustomerApp-1.0-SNAPSHOT.war \**

**"http://54.145.142.96:8080/manager/text/deploy?path=/simplecustomerapp&update=true"**

### **Step 6: Slack Notification**

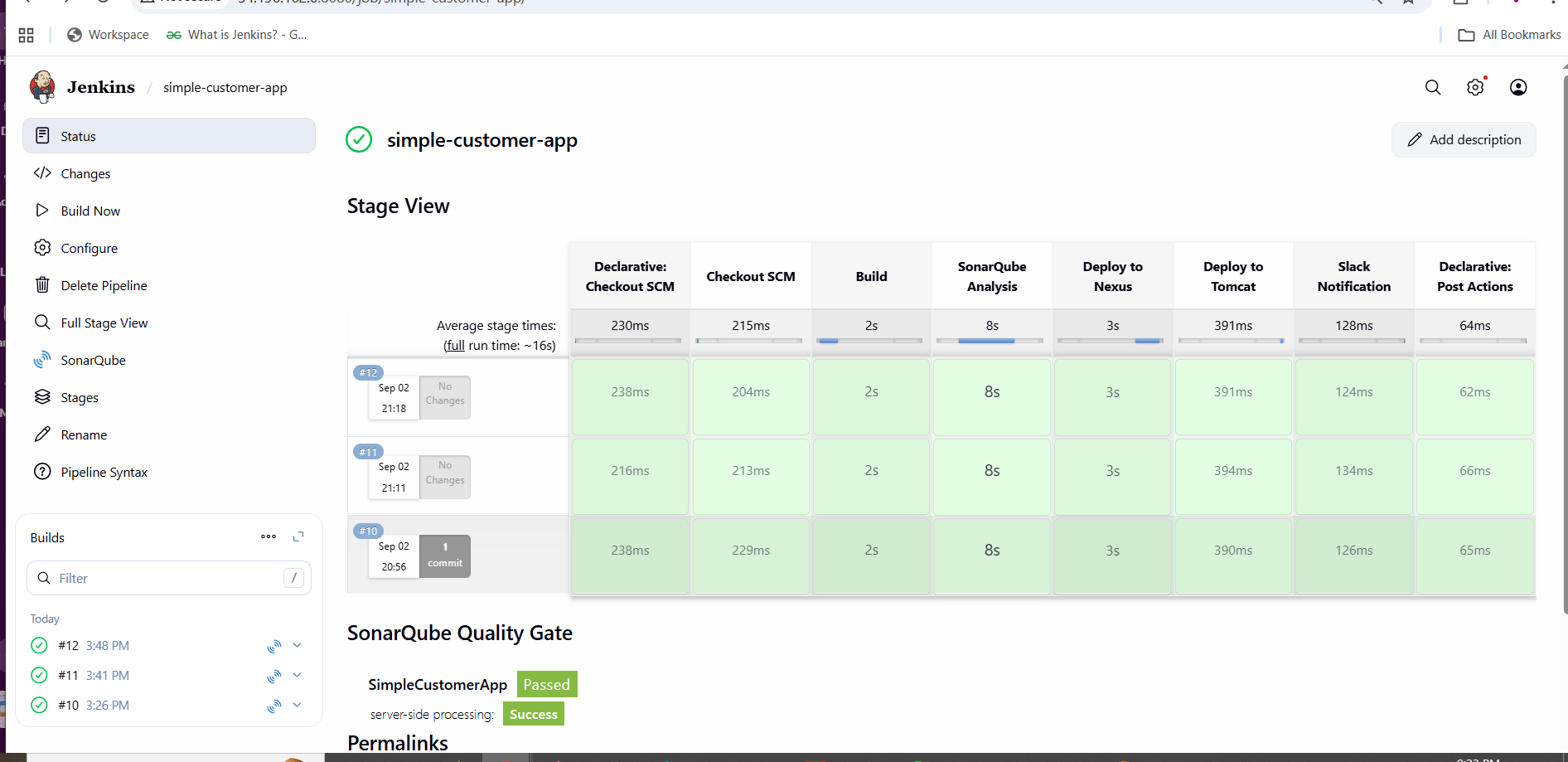
* **Build success/failure notifications sent to #jenkins-integration.**

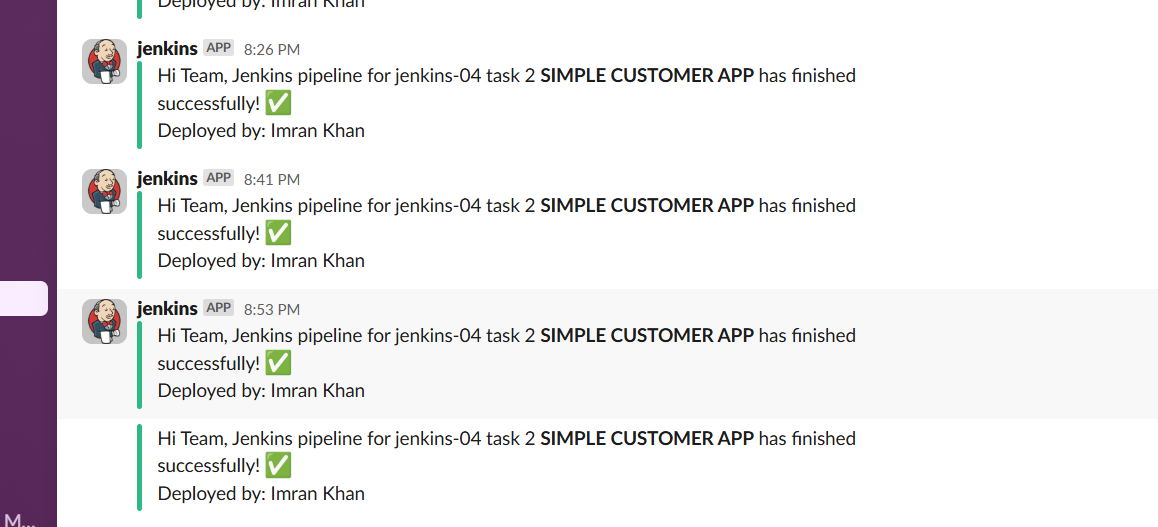
## **6. Post Deployment Verification**

1. **Check Tomcat webapps folder:**

**ls -l /opt/tomcat/webapps**

[**http://54.145.142.96:8080/simplecustomerapp**](http://54.145.142.96:8080/simplecustomerapp)

****

****

**3) Setup a jenkins CICD pipeline using Scripted pipeline using feature-1.1 branch.**

[**https://github.com/betawins/sabear\_simplecutomerapp/tree/feature-1.1**](https://github.com/betawins/sabear_simplecutomerapp/tree/feature-1.1)

stages:

1) Git Clone

2) Sonarqube Integration

3) Maven Compilation

4) Nexus Artifactory

5) Slack Notification

6) Deploy On tomcat

# **1) Prereqs you set up**

* **Jenkins controller** with plugins: *Pipeline, Git, Maven Integration, SonarQube Scanner, Credentials, Slack*.
* **Tools in Jenkins → Global Tool Config**
  + JDK (e.g., Java 17)
  + Maven named **Maven-3.8.4**
* **Servers**
  + **SonarQube** reachable and configured in Jenkins as server name **SonarQube**.
  + **Nexus** (two repos: *maven-snapshots* and *releases*).
  + **Tomcat** with *manager-script* user enabled.
* **Credentials in Jenkins → Credentials**
  + nexus-creds (username/password)
  + tomcat-credentials (username/password)
  + slack\_notification (Slack token for your workspace)
* **Maven settings on the Jenkins box**: /var/lib/jenkins/.m2/settings.xml with matching <server> IDs for your POM:

**<distributionManagement>**

**<repository>**

**<id>releases</id>**

**<url>http://54.209.186.138:8081/repository/maven-releases/</url>**

**</repository>**

**<snapshotRepository>**

**<id>snapshots</id>**

**<url>http://54.209.186.138:8081/repository/maven-snapshots/</url>**

**</snapshotRepository>**

**</distributionManagement>**

# **2) The Scripted Pipeline you ran**

**Repo & branch:** https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git @ **feature-1.1**

**node {**

**stage('Git Clone') {**

**git branch: 'feature-1.1',**

**url: 'https://github.com/imrankhanmohammad257/sabear\_simplecutomerapp.git'**

**}**

**stage('SonarQube Analysis') {**

**withSonarQubeEnv('SonarQube') {**

**sh 'mvn sonar:sonar'**

**}**

**}**

**stage('Maven Compilation') {**

**def mvnHome = tool name: 'Maven-3.8.4', type: 'maven'**

**sh "${mvnHome}/bin/mvn clean compile"**

**}**

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

**withCredentials([usernamePassword(credentialsId: 'nexus-creds', usernameVariable: 'NEXUS\_USER', passwordVariable: 'NEXUS\_PASS')]) {**

**sh '''**

**mvn clean deploy -DskipTests --settings /var/lib/jenkins/.m2/settings.xml**

**'''**

**}**

**}**

**stage('Slack Notification') {**

**slackSend(**

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

**color: 'good',**

**message: "Jenkins Scripted Pipeline job for \*sabear\_simplecutomerapp (feature-1.1)\* finished successfully! ✅"**

**)**

**}**

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

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

**sh '''**

**set -e**

**WAR\_FILE=$(ls target/\*.war | head -n 1)**

**echo "WAR file to deploy: $WAR\_FILE"**

**echo "Undeploying old app..."**

**curl -s -o /dev/null -w "%{http\_code}\\n" -u $TOMCAT\_USER:$TOMCAT\_PASS \**

**"http://54.87.222.232:8080/manager/text/undeploy?path=/customerapp"**

**echo "Deploying new WAR..."**

**curl -s -o /dev/null -w "%{http\_code}\\n" -u $TOMCAT\_USER:$TOMCAT\_PASS \**

**-T "$WAR\_FILE" \**

**"http://54.87.222.232:8080/manager/text/deploy?path=/customerapp&update=true"**

**echo "Deployment completed."**

**'''**

**}**

**}**

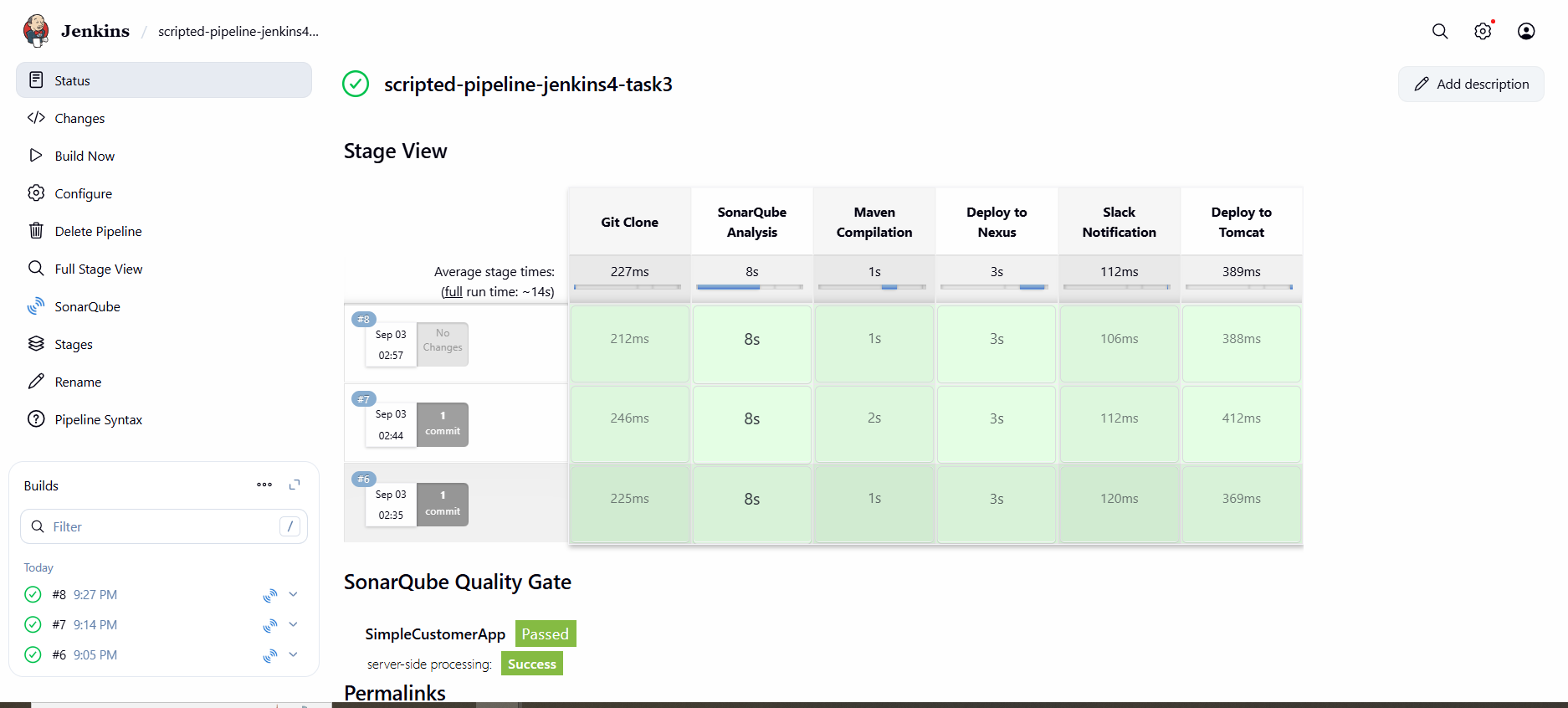
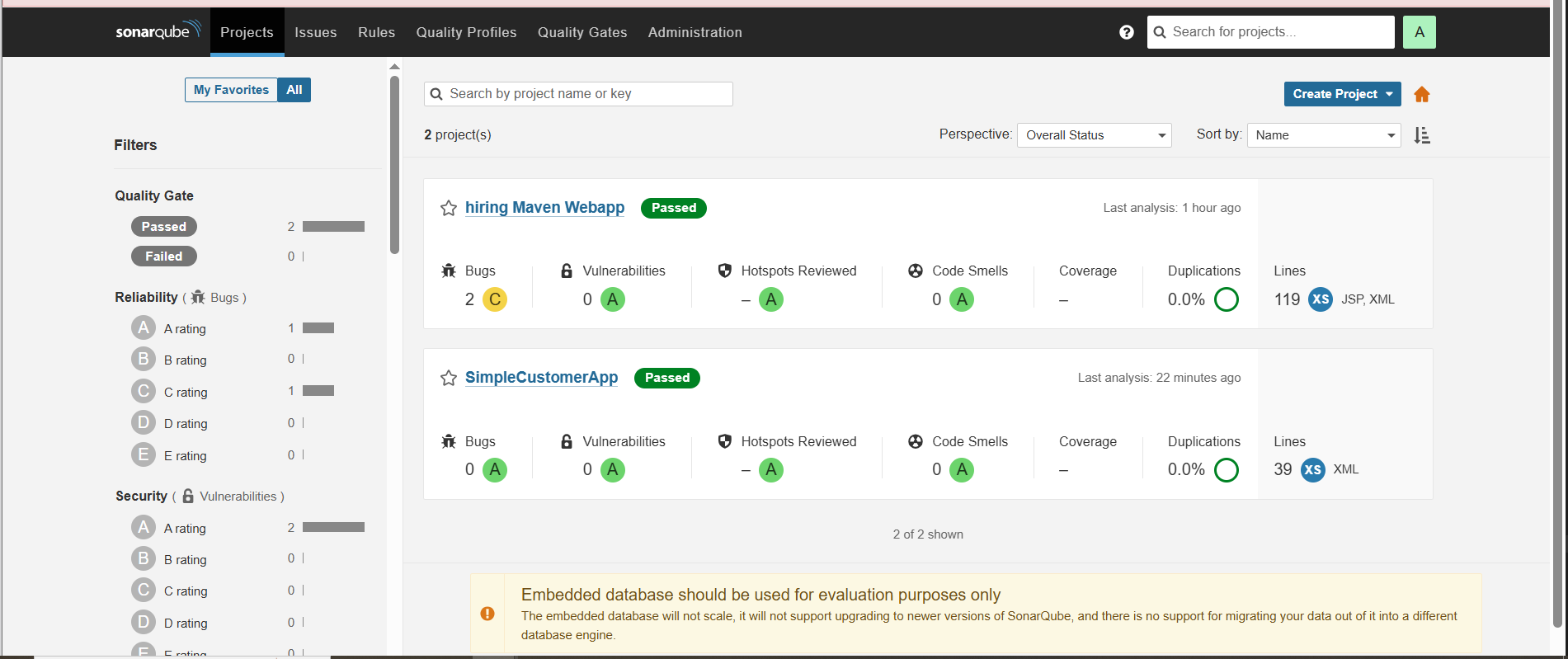
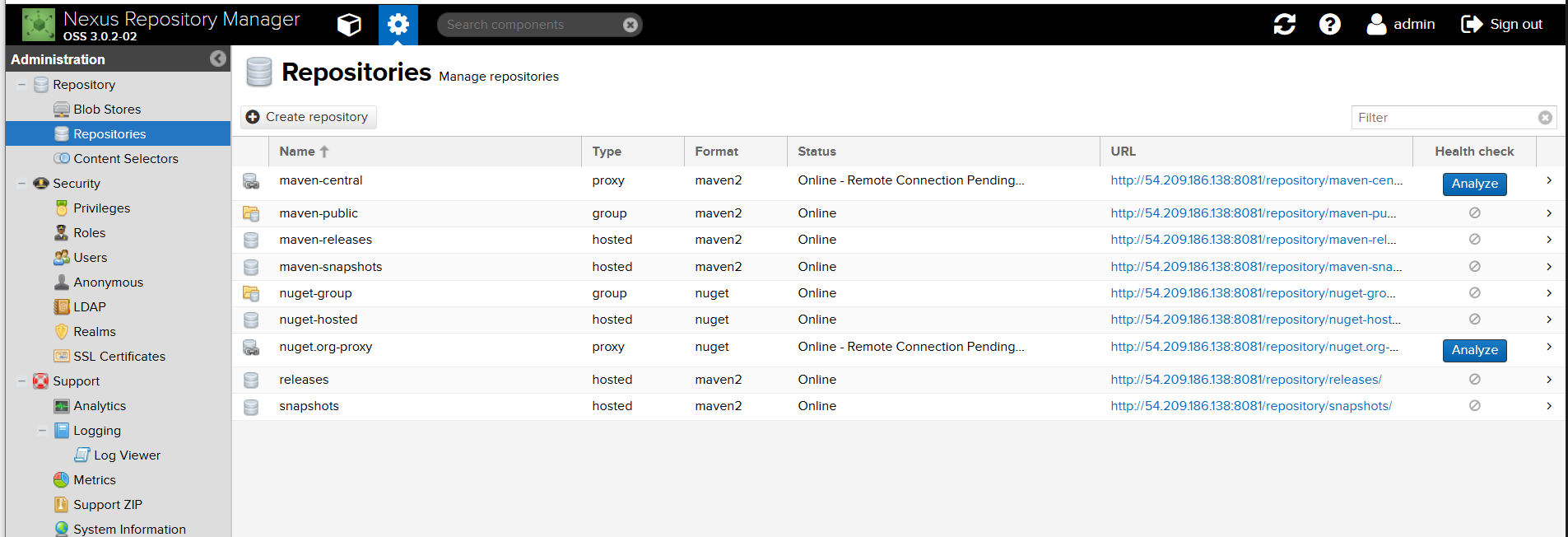
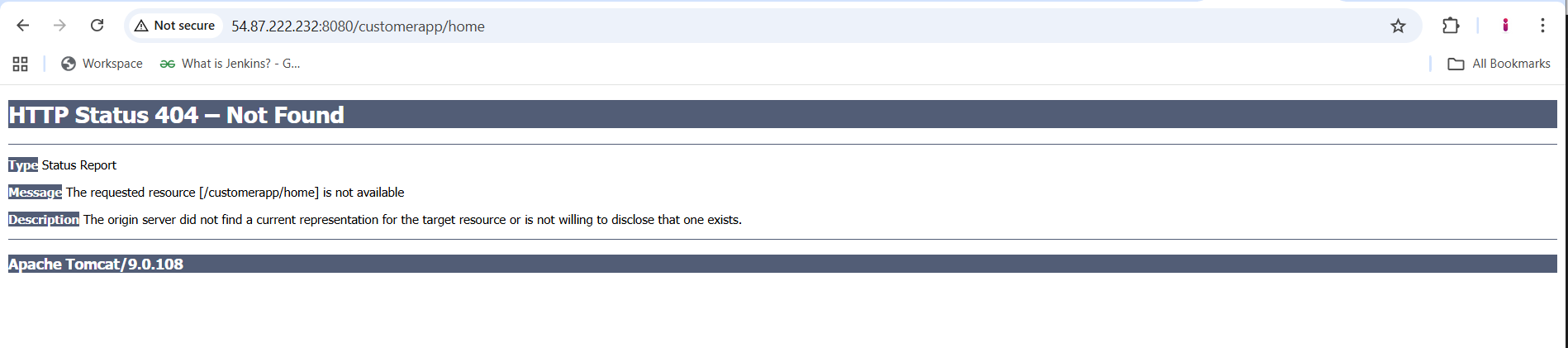
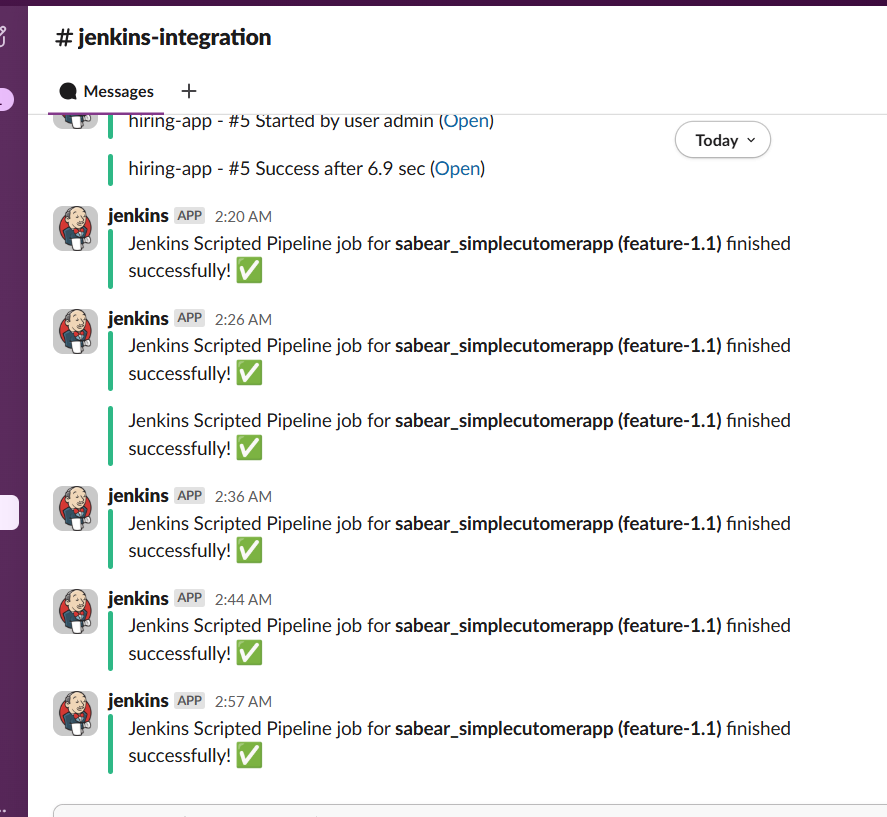
**}**

# **3) What happened in each stage (what you observed)**

1. **Git Clone  
    Jenkins pulled feature-1.1 into the job workspace.**
2. **Maven Build/Package**
   * **Executed mvn clean package -DskipTests.**
   * **Produced target/SimpleCustomerApp-1.0-SNAPSHOT.war.**
3. **SonarQube Analysis**
   * **Ran mvn sonar:sonar under withSonarQubeEnv('SonarQube').**
   * **SonarQube accepted the report (dashboard link showed analysis OK).**
4. **Nexus Deploy**
   * **Attempted mvn deploy.**
   * **You initially hit issues:**
     + **401 Unauthorized → fixed by ensuring settings.xml <servers> IDs matched your POM’s <distributionManagement> IDs and by using Jenkins credentials.**
     + **Connection timed out → pointed Maven to the correct Nexus IP/port.**
   * **After fixes, artifacts upload to snapshots repo (because version is 1.0-SNAPSHOT).**
5. **Deploy on Tomcat**
   * **You added an undeploy first call to avoid stale app:  
      .../manager/text/undeploy?path=/customerapp**
   * **Then deployed the freshly built WAR to context /customerapp with update=true.**
   * **Tomcat logs (tail -f /opt/tomcat/logs/catalina.out) confirmed (un)deploy events.**
6. **Slack Notification**
   * **Sent status messages to #jenkins-integration using slack\_notification token.**

# **4) Troubleshooting you performed**

* **Locked target/ causing clean failures → added rm -rf target before build.**
* **Nexus 401 → aligned <server id> names in settings.xml with POM IDs; used Jenkins creds.**
* **Nexus timeout → corrected the Nexus endpoint IP/port.**
* **Tomcat showing old content → added explicit undeploy step before deploy.**
* **Nothing visible at /customerapp/ → inspected the WAR:**
  + **jar tf SimpleCustomerApp-1.0-SNAPSHOT.war showed no JSP/HTML and no web.xml.**
  + **Root cause: the repo’s src/main/webapp/ is empty, so there’s nothing to render.**
  + **Next action (outside pipeline): add an index.jsp or a servlet + web.xml, rebuild, redeploy.**

1. **BUILD**
2. ****
3. **SONARQUBE**
4. ****
5. ****
6. **NEXUS**
7. ****
8. **TOMCAT**
9. ****
10. **SLACK NOTIFICATION**
11. ****

**4) Write sample skeleton of pipelines.**

**Jenkins supports two primary pipeline syntaxes: Declarative and Scripted. Here's an overview of both:**

### **a) Declarative Pipeline**

**This is a more structured and opinionated syntax, making it easier to read and maintain.**

**This structure is ideal for most use cases and is recommended for new users.**

****

### **b) Scripted Pipeline**

This offers more flexibility and control, using Groovy scripting.

While powerful, it requires a deeper understanding of Groovy and Jenkins internals.

****

**5) Create a parametirized job in jenkins.**

[**https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git**](https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git)

## **Step 1: Log in to Jenkins**

1. **Open your Jenkins URL in a browser:**

**http://<JENKINS\_SERVER\_IP>:8080**

## **Step 2: Create a New Job**

1. **Click “New Item” in Jenkins dashboard.**
2. **Enter a name for your job, e.g., Spring3-MVC-Parameterized.**
3. **Select “Freestyle project” or “Pipeline” (for this example, we will use Pipeline).**
4. **Click OK.**

## **Step 3: Enable Parameterization**

1. **In the job configuration, check “This project is parameterized”.**
2. **Click “Add Parameter” and select String Parameter.**
3. **Configure the parameter:**
   * **Name: BRANCH\_NAME (used to choose Git branch)**
   * **Default Value: main**
   * **Description: Enter the Git branch to build**
4. **Add another parameter if needed:**
   * **Example: DEPLOY\_ENV (to choose environment like dev, test, prod)**

## **Step 4: Configure Pipeline**

1. **Scroll to Pipeline section.**
2. **Select Pipeline script or Pipeline script from SCM:**
   * **SCM: Git**
   * **Repository URL:**

**https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git**

* + **Branch Specifier: ${BRANCH\_NAME} (uses the parameter)**
  + **Script Path: Jenkinsfile (if using Pipeline script from repo)**

## **Step 5: Add Build Stages**

**If using Pipeline Script directly in Jenkins:**

**pipeline {**

**agent any**

**tools {**

**maven "Maven-3.8.4"**

**}**

**environment {**

**NEXUS\_URL = "54.163.17.174:8081"**

**NEXUS\_REPOSITORY = "releases"**

**NEXUS\_CREDENTIAL\_ID = "nexus-creds"**

**TOMCAT\_USER = "deployer" // Tomcat manager username**

**TOMCAT\_PASSWORD = "deployer" // Tomcat manager password**

**TOMCAT\_HOST = "3.82.42.125" // Tomcat EC2 public IP**

**TOMCAT\_PORT = "8080"**

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

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

**}**

**stages {**

**stage("Clone code") {**

**steps {**

**git 'https://github.com/imrankhanmohammad257/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 artifact: ${warFile} (version: ${artifactVersion}) to Nexus"**

**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>"**

**)**

**cleanWs()**

**}**

**failure {**

**slackSend(**

**channel: SLACK\_CHANNEL,**

**color: 'danger',**

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

**)**

**cleanWs()**

**}**

**always {**

**echo "Cleaning workspace..."**

**cleanWs()**

**}**

**}**

**}**

## **Step 6: Save and Run Job**

1. Click **Save**.
2. Click **Build with Parameters**.
3. Enter values for parameters:  
   * BRANCH\_NAME: e.g., main or any branch in Git repo
   * DEPLOY\_ENV: e.g., dev
4. Click **Build**.

## **Step 7: Verify WAR Artifact**

1. On Jenkins, navigate to job workspace:

/var/lib/jenkins/workspace/Spring3-MVC-Parameterized/target/

## **Step 8: Deploy to Tomcat**

1. Copy WAR file to Tomcat EC2:

scp /var/lib/jenkins/workspace/Spring3-MVC-Parameterized/target/hello-world.war ec2-user@<TOMCAT\_IP>:/opt/tomcat/webapps/

tail -f /opt/tomcat/logs/catalina.out

http://<TOMCAT\_IP>:8080/hello-world/

## **Step 9: Add Slack Notifications (Optional)**

1. Install **Slack Notification Plugin** in Jenkins.
2. Configure Slack credentials in **Manage Jenkins → Configure System**.
3. Add in Pipeline post section:

post {

success {

slackSend(channel: '#jenkins', color: 'good', message: "✅ Build Successful: ${env.JOB\_NAME} #${env.BUILD\_NUMBER}")

}

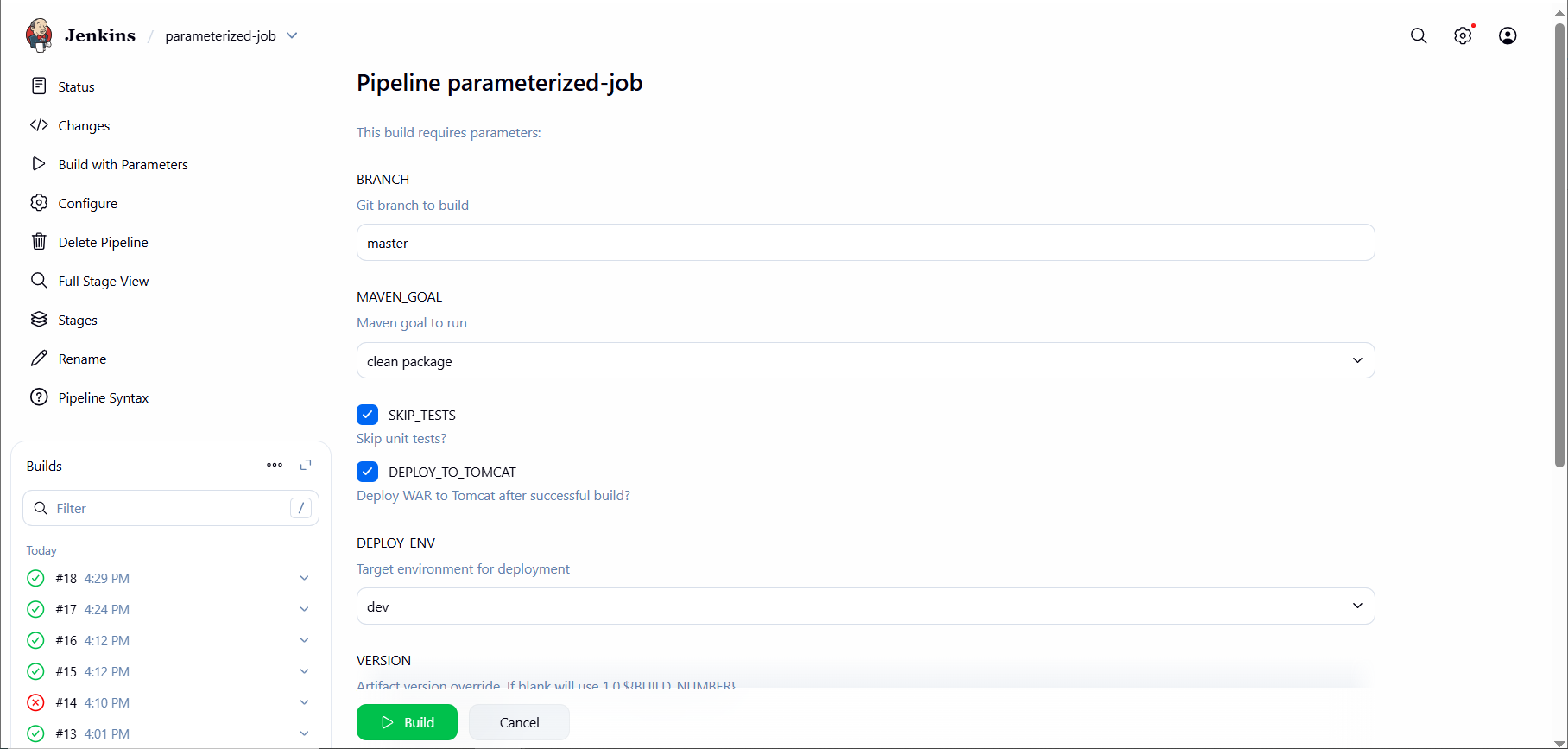
failure {

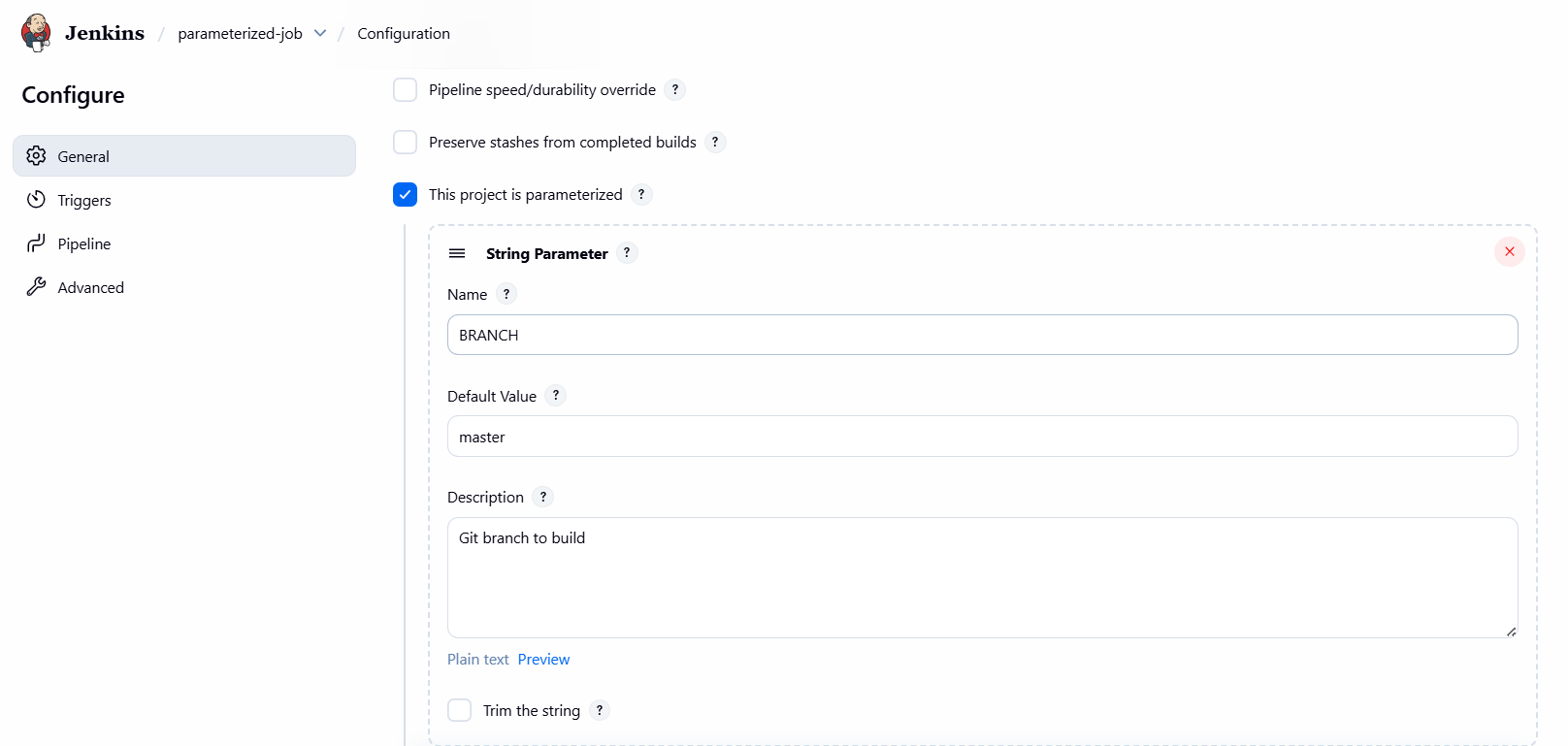
slackSend(channel: '#jenkins', color: 'danger', message: "❌ Build Failed: ${env.JOB\_NAME} #${env.BUILD\_NUMBER}")

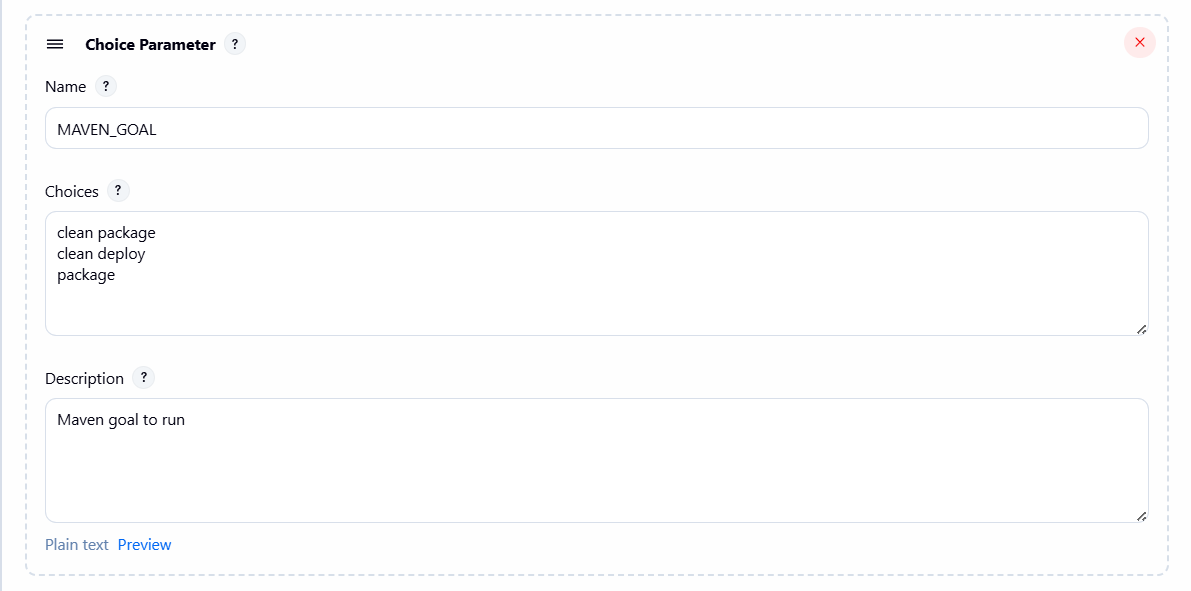
}

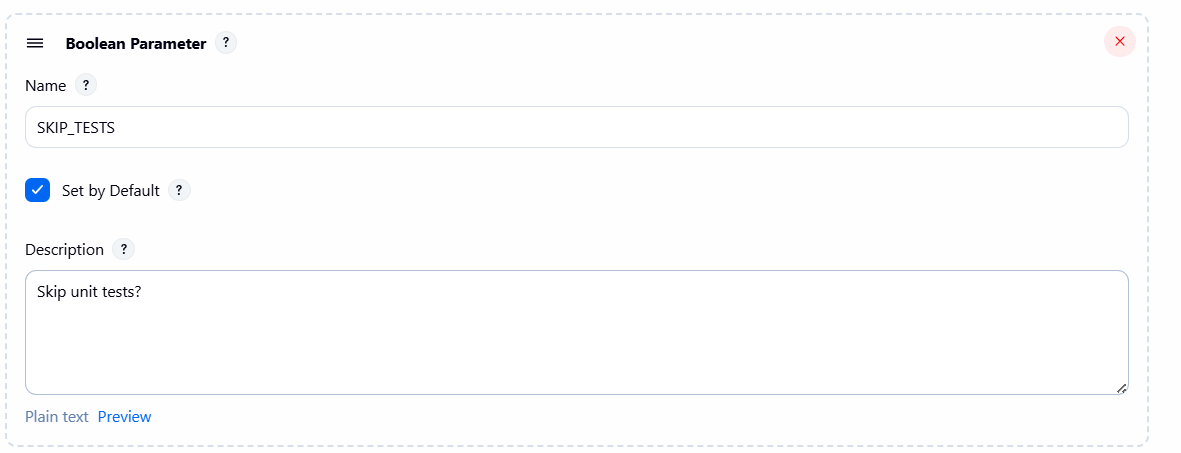
}

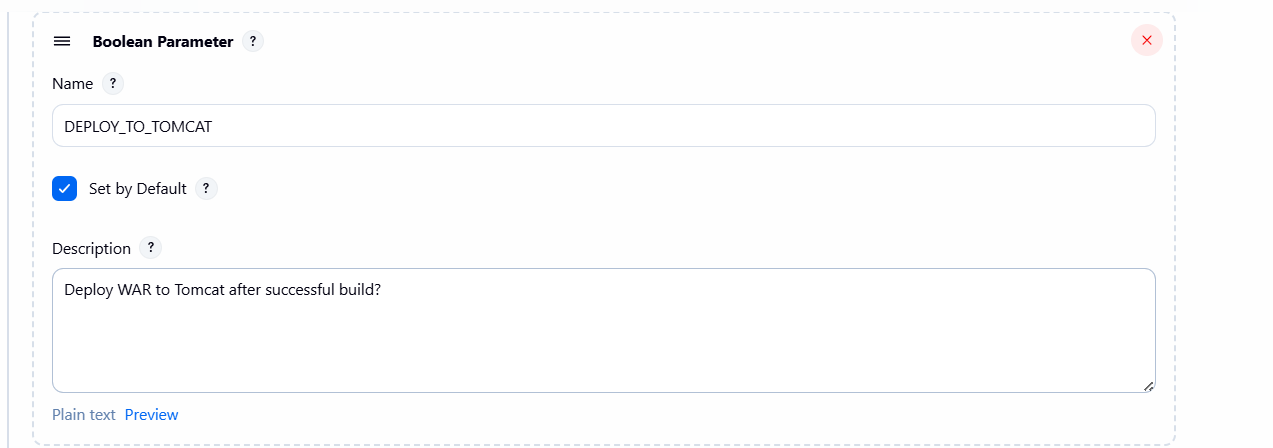
**JENKINS**

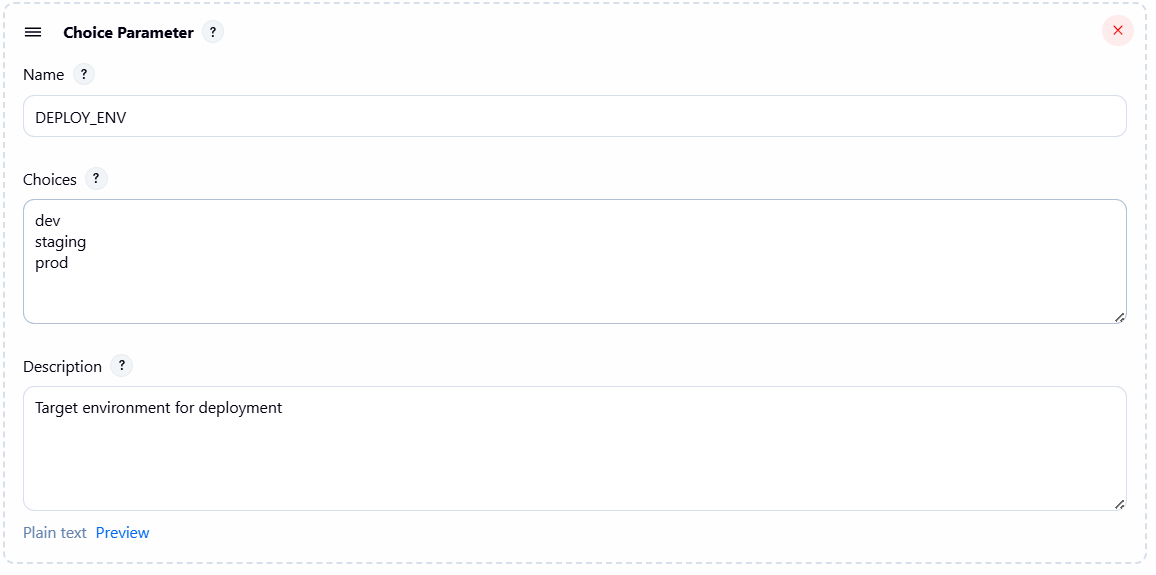
****

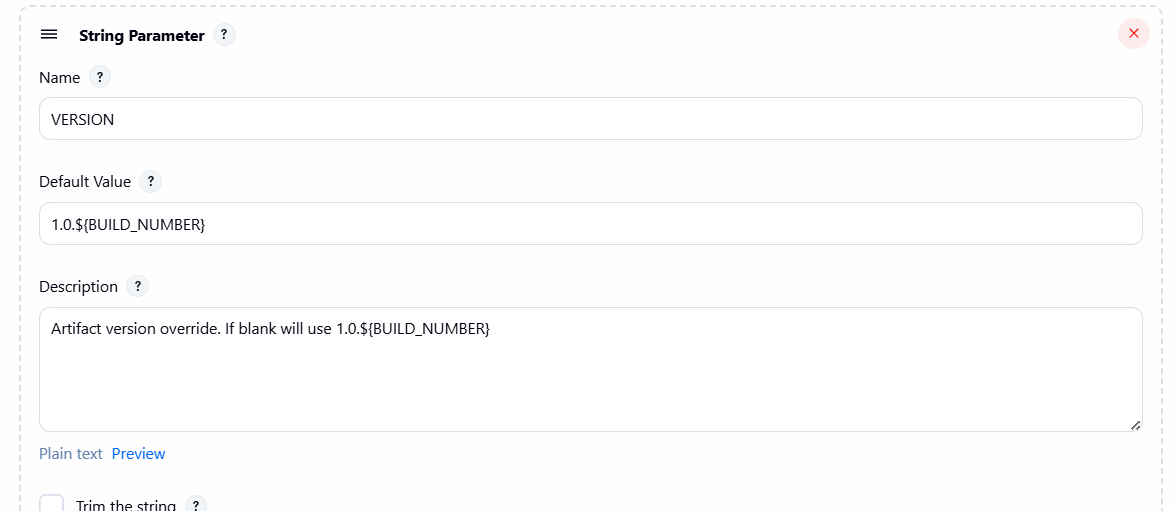
****

****

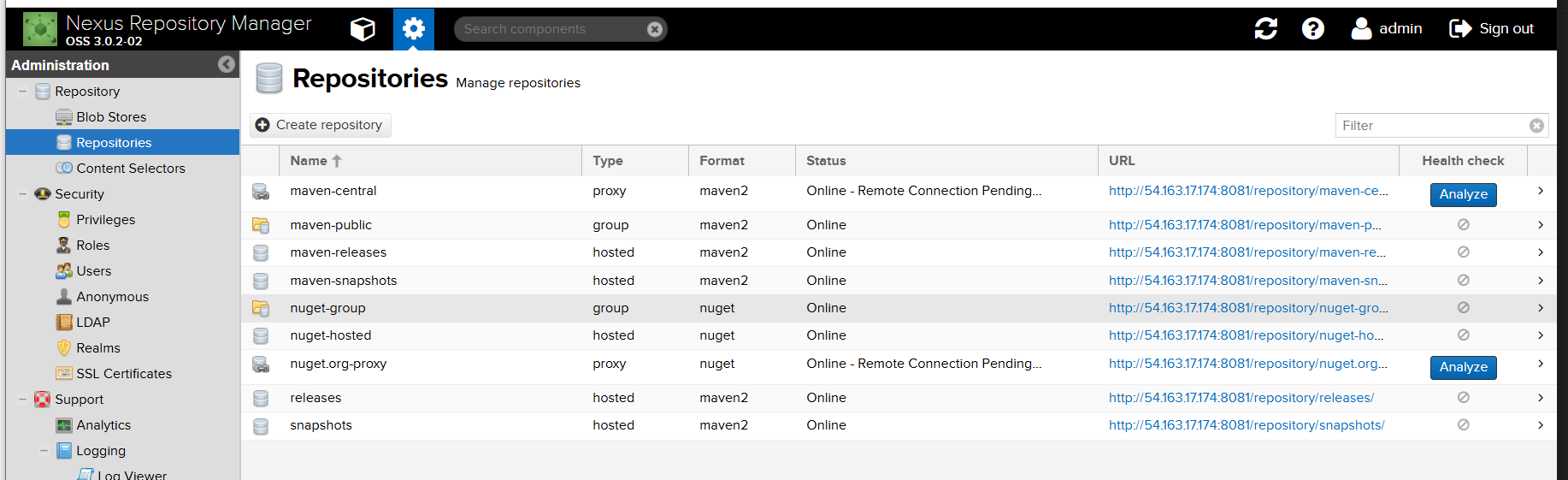
****

****

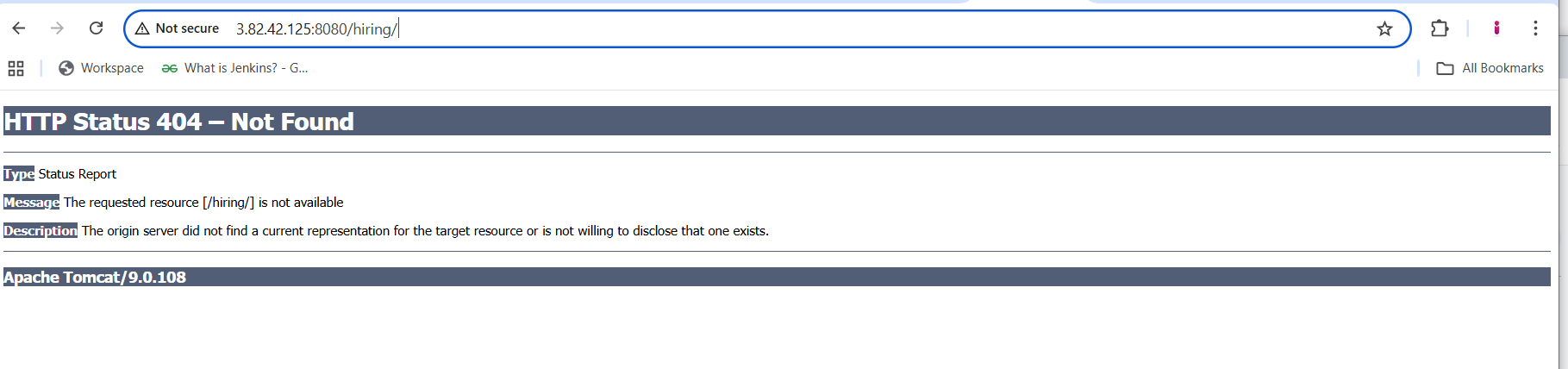
****

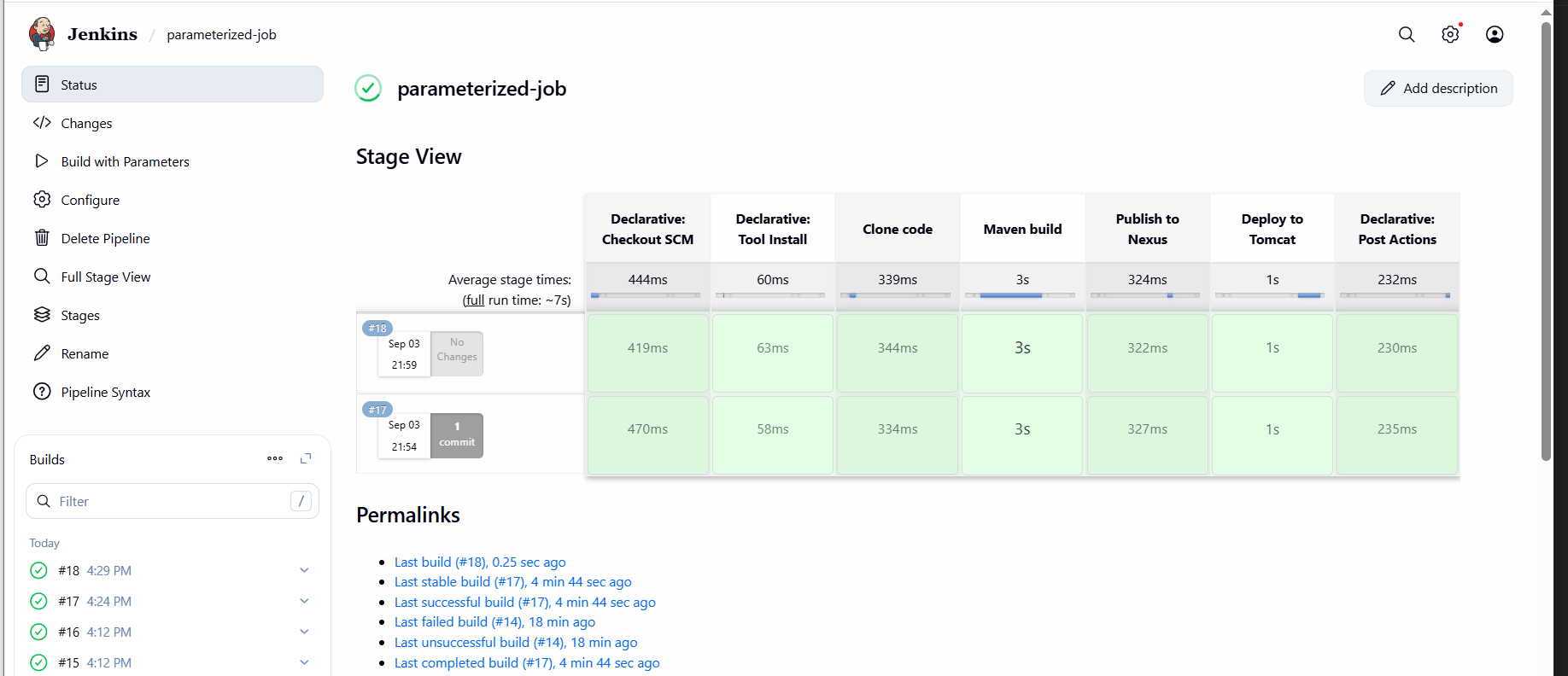
****

**NEXUS**

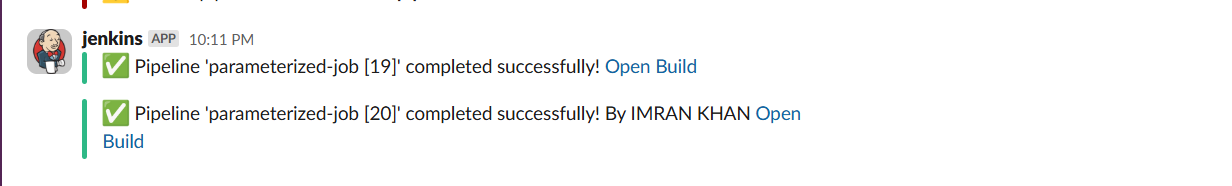
****

**TOMCAT**

****

****

**SLACK**

****

**6) Setup one slave machine for jenkins.**

[**GitHub - betawins/hiring-app**](https://github.com/betawins/hiring-app.git)

[**GitHub - betawins/sabear\_simplecutomerapp at feature-1.1**](https://github.com/betawins/sabear_simplecutomerapp/tree/feature-1.1)

[**GitHub - betawins/spring3-mvc-maven-xml-hello-world-1: Maven + Spring 3 MVC hello world example (XML)**](https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git)

1.

1. **Create jenkins ec2 installed java , git and jenkins.**
2. **Create or launch instances slave1-ec2 Installed java and git on slave1-ec2(jenkins no need in slave1-ec2)**
3. **sudo yum install -y git**
4. **sudo dnf install -y java-17-amazon-corretto**
5. **copy the key gen of slave1-ec2 → ssh-keygen**
6. **cd .ssh**
7. **cat id\_rsa.pub > authorized\_keys**
8. **chmod 700 authorized\_keys**
9. **In root → mkdir slave1-workspace**

### **Steps on Master Machine**

1. **Login to master machine.**
2. **Switch to root user.**
3. **Create Jenkins SSH directory:**

**mkdir -p /var/lib/jenkins/.ssh**

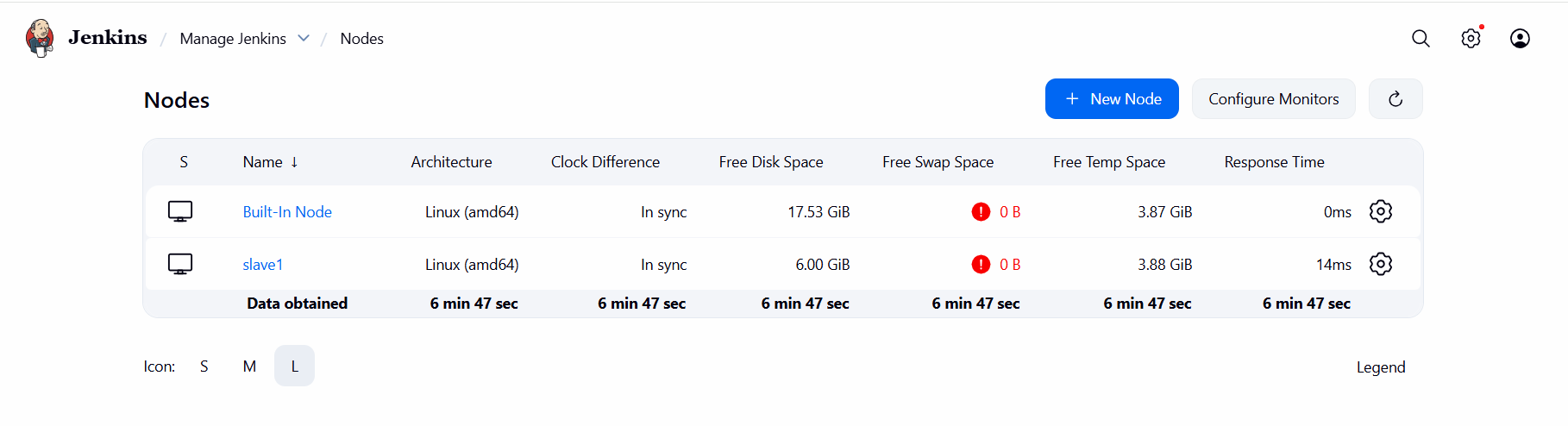
**cd /var/lib/jenkins/.ssh**

1. **ssh-keyscan -H SLAVE-NODE-PUBLIC-IP >> /var/lib/jenkins/.ssh/known\_hosts**
2. **chown jenkins:jenkins /var/lib/jenkins/.ssh/known\_hosts**
3. **chmod 644 /var/lib/jenkins/.ssh/known\_hosts**

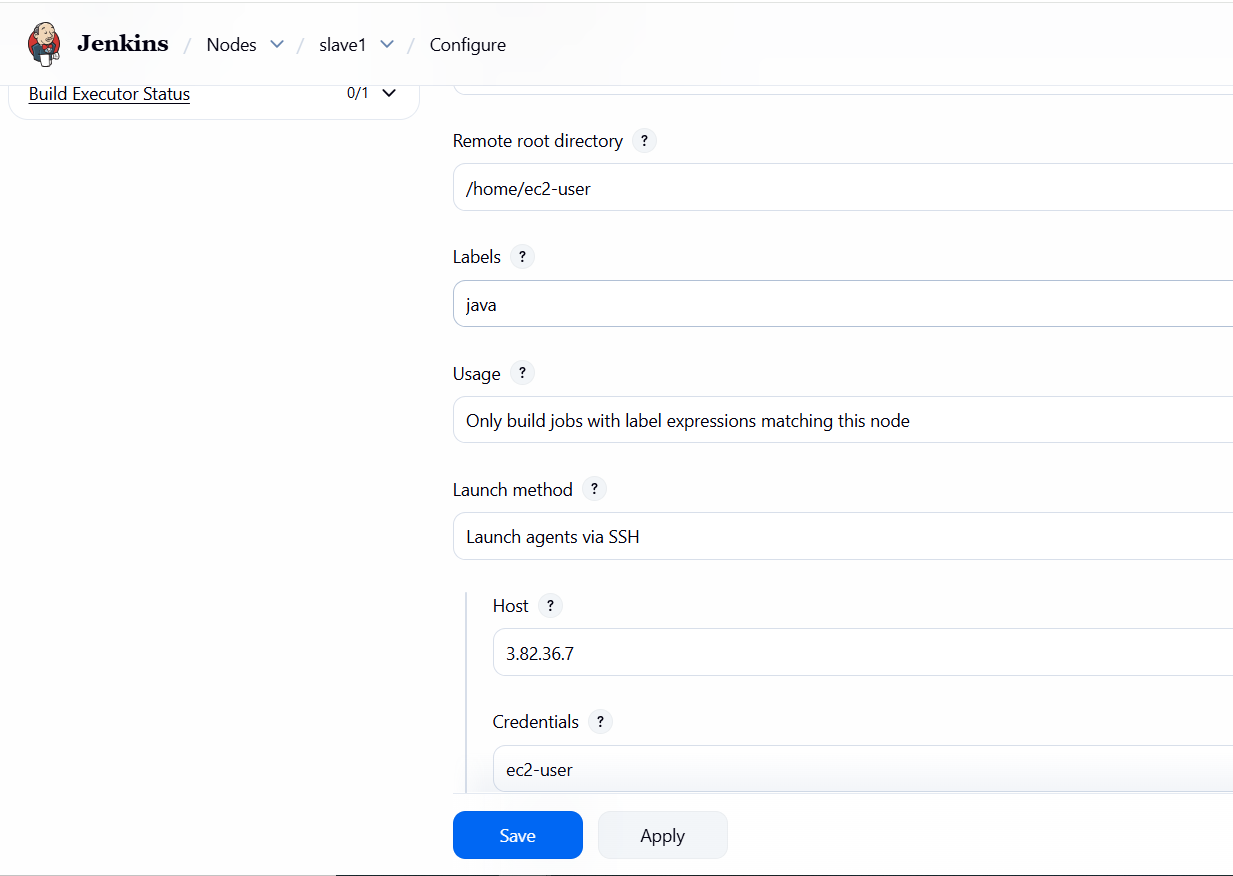
### **E) Create the node in Jenkins GUI**

1. **Manage Jenkins → Nodes → New Node**
2. **Name: slave1 → Type: Permanent Agent**
3. **Remote root directory: /home/ec2-user**
4. **This path is taken from slave machine**
5. **Slave ec2 → cat /etc/passwd**
6. **Create mkdir slave1-workspace(if already created then ignore)**
7. **In slave root → chown ec2-user:ec2-user slave1-workspace**
8. **chmod 777 slave1-workspace**
9. **Labels: slave1 java (as you like)**
10. **Usage: “Only build jobs with label expressions” (optional)**
11. **Launch method: Launch agents via SSH**
    * **Host: <NEW\_AGENT\_IP\_OR\_DNS>**
    * **Credentials: create/select “SSH Username with private key”**
      + **Username: ec2-user**
      + **Private key: paste content of /var/lib/jenkins/.ssh/id\_rsa from the master or pem key.**
    * **Host Key Verification Strategy: Known hosts file (recommended)**
12. **Save → Jenkins should connect and show Online.**

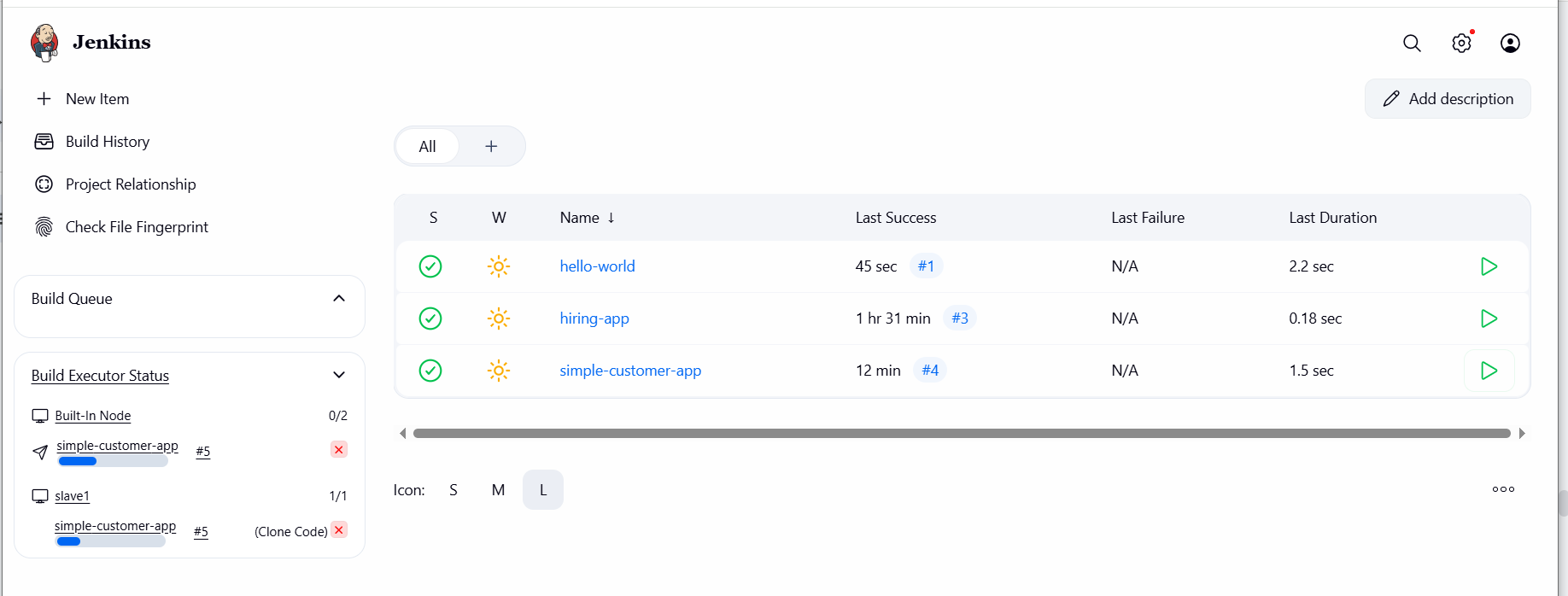
**JENKINS NODE**

****

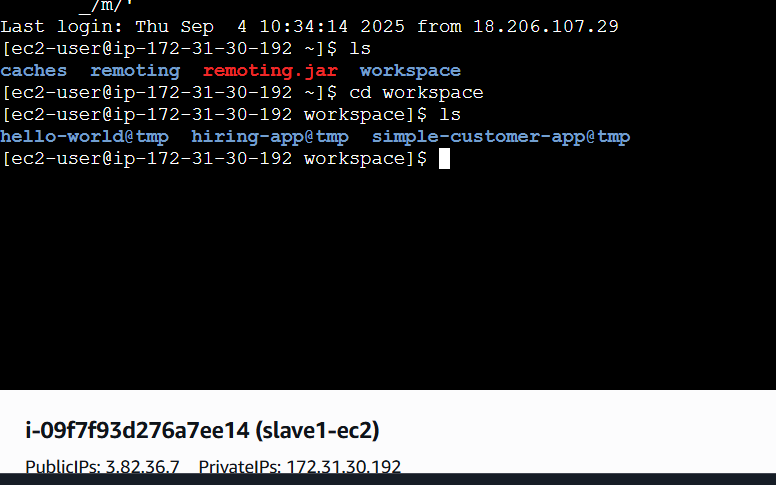
**CONFIGURE SLAVE NODE**

****

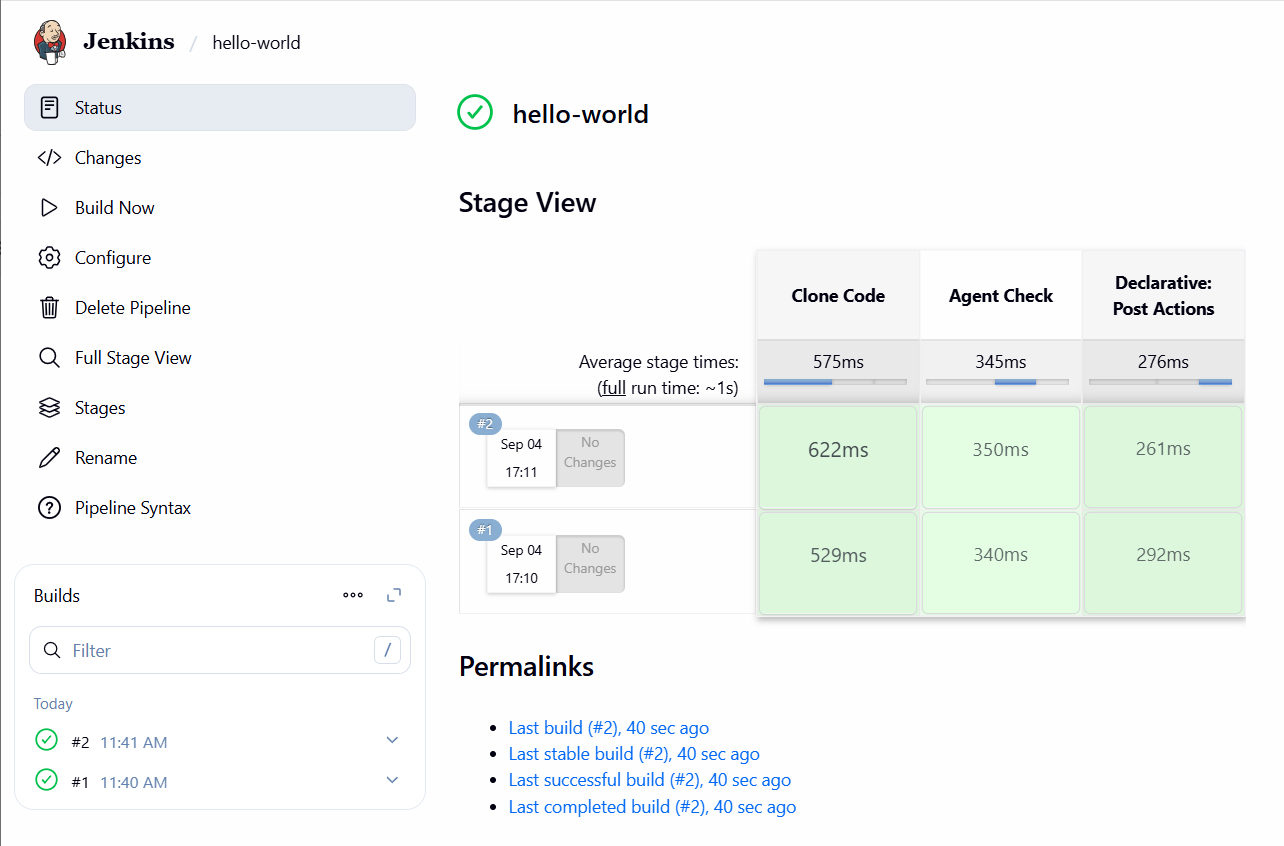
**JOBS Running on Master and Slave1**

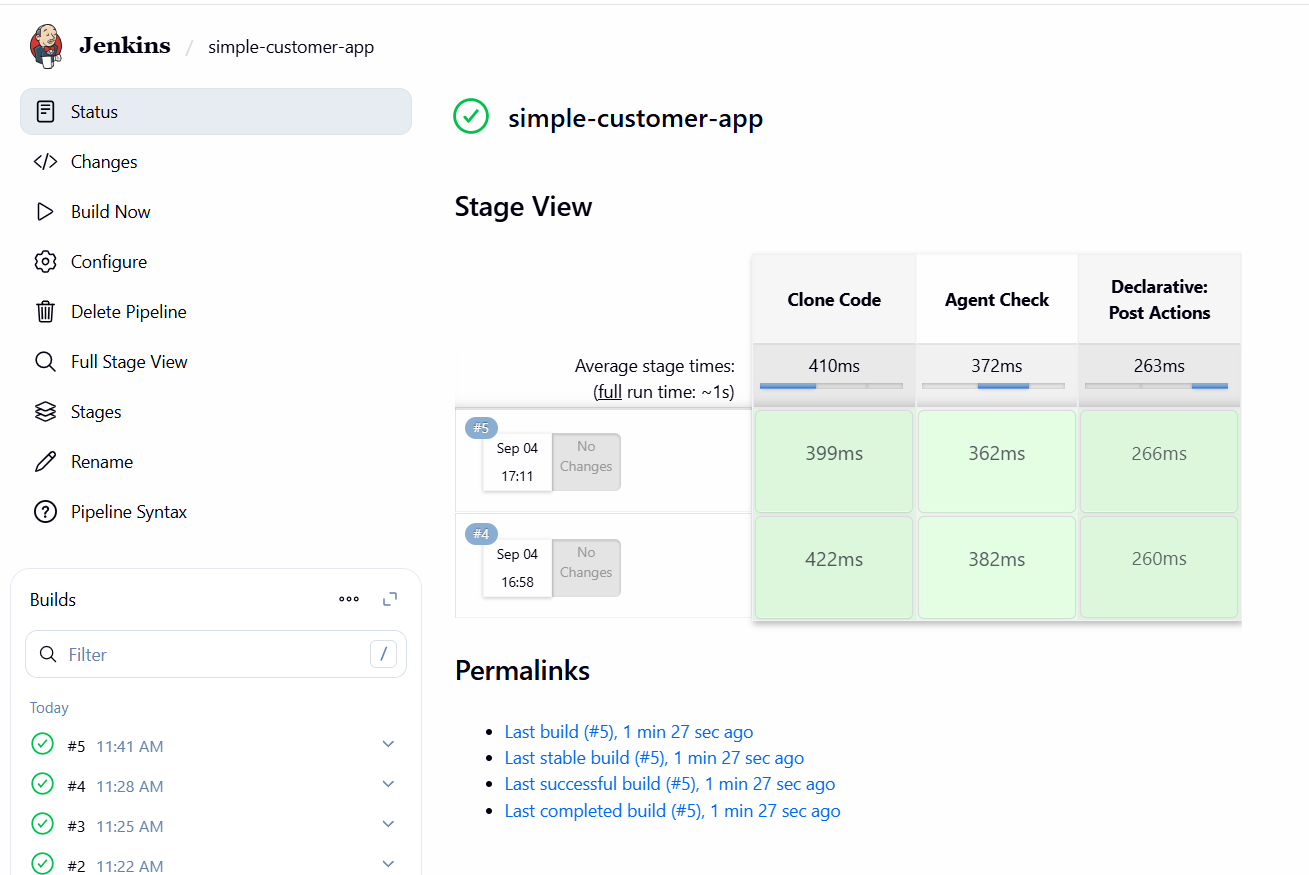


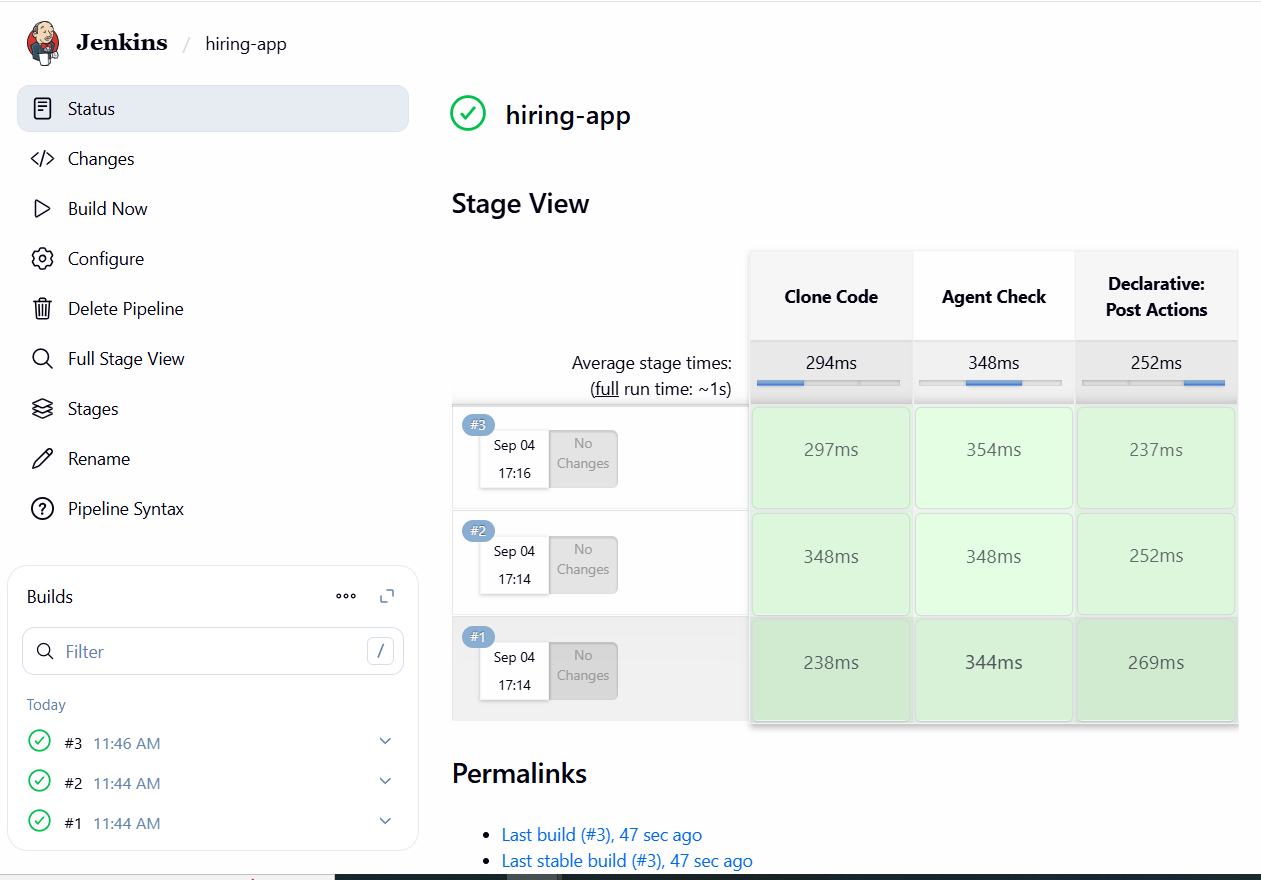
**On Slave EC2 workspace , after build files are created.**



**BUILD STATUS OF DIFFERENT JOBS**







**SLACK NOTIFICATION**

