Jenkins Pipeline

--> It is a way to define CI CD process as a code

--> CI CD workflow will be defined as a code in Jenkins pipeline and whenever we are dealing with complex CI/CD process it is highly recommended to go with Jenkins pipeline concept

Pipeline contains stages to perform CI CD

--> Clone Git Repo

--> Build that project with Maven

--> Review the code with tools (SonarQube)

--> Artifact upload using Nexus

--> Build Docker image

--> Add into the Docker hub

--> Push Docker image into registry

--> Deploy App in K8s

We are going to automate the entire CI/CD pipeline using two approaches:

1. Declarative pipeline

pipeline {

agents any ---> where we want to execute our job

tools {

maven “maven.3.9”

}

stages {

// 3 operations I am performing

// cloning the code

// building the project

// reviewing the code

// uploading using Nexus etc

stage (‘Git Clone’){

steps {

echo ‘cloning git repo…’

}

}

stage (‘Maven Build’){

steps {

echo ‘project build with Maven’

}

}

stage (‘Deploy’){

steps {

echo ‘deploying application with Tomcat’

}

}

}

}

Declarative pipeline with Jenkins + Git + Maven

pipeline {

agent any

tools {

maven "maven-3.9.10"

}

stages {

stage('git clone') {

steps {

git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

}

}

stage('maven build') {

steps {

sh 'mvn clean compile test package'

}

}

stage('deploy') {

steps {

echo 'Deploying App with Tomcat'

}

}

}

}

Declarative pipeline with Jenkins + Git + Maven + Tomcat

---> SSH Agent Configuration

It is used to establish remote SSH connection from one server (Linux VM) to another server (Linux VM)

Ex: Jenkins server is getting connected to Tomcat server to copy WAR file

pipeline {

agent any

tools {

maven "maven-3.9.10"

}

stages {

stage('git clone') {

steps {

git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

}

}

stage('maven build') {

steps {

sh 'mvn clean compile test package'

}

}

stage('App deployment') {

steps {

sshagent(['Tomcat-Server-Credentials-Pipeline']) {

sh 'scp -o StrictHostKeyChecking=no target/FirstSpringWebApp-0.0.1-SNAPSHOT.war ec2-user@15.156.94.232:/home/ec2-user/apache-tomcat-11.0.8/webapps'

}

}

}

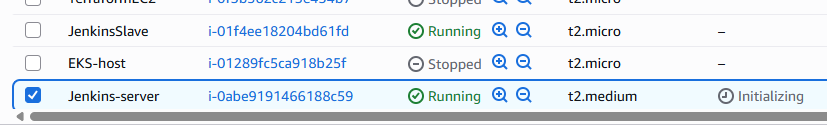
}

}

1. Scripted pipeline (Groovy)

22.00

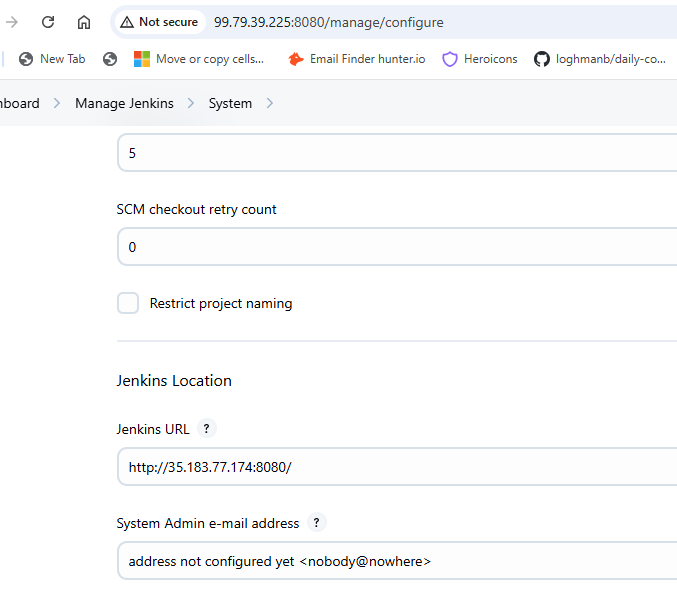
Start JenkinsServer and Slave



Open

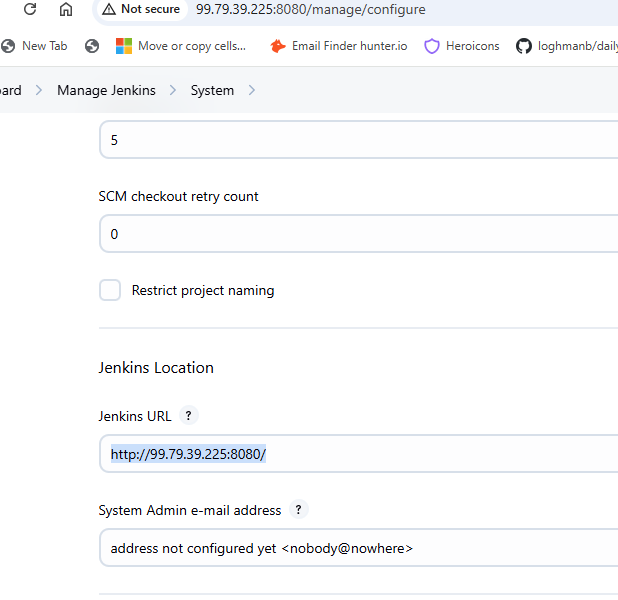
<http://99.79.39.225:8080/manage/configure>

What we have here is different from public IP



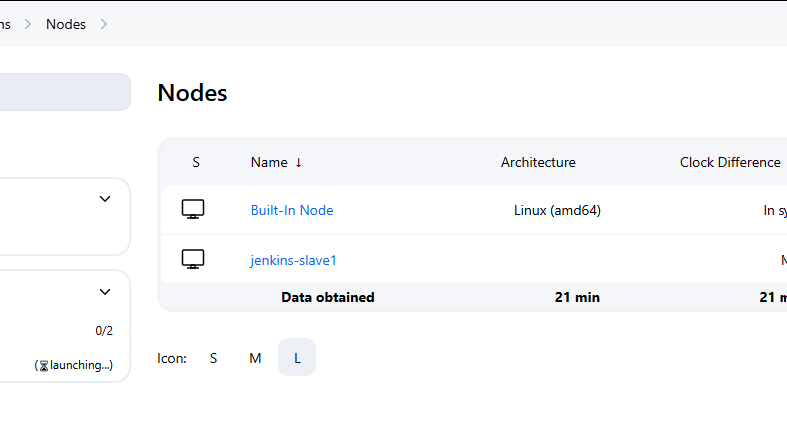
I update Jenkins URL

<http://99.79.39.225:8080/>

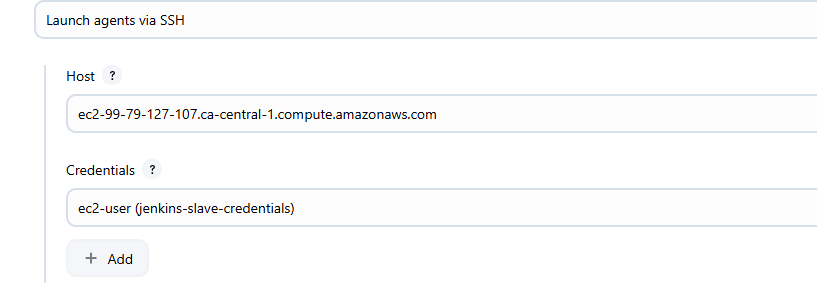


Apply and Save

Go to Nodes



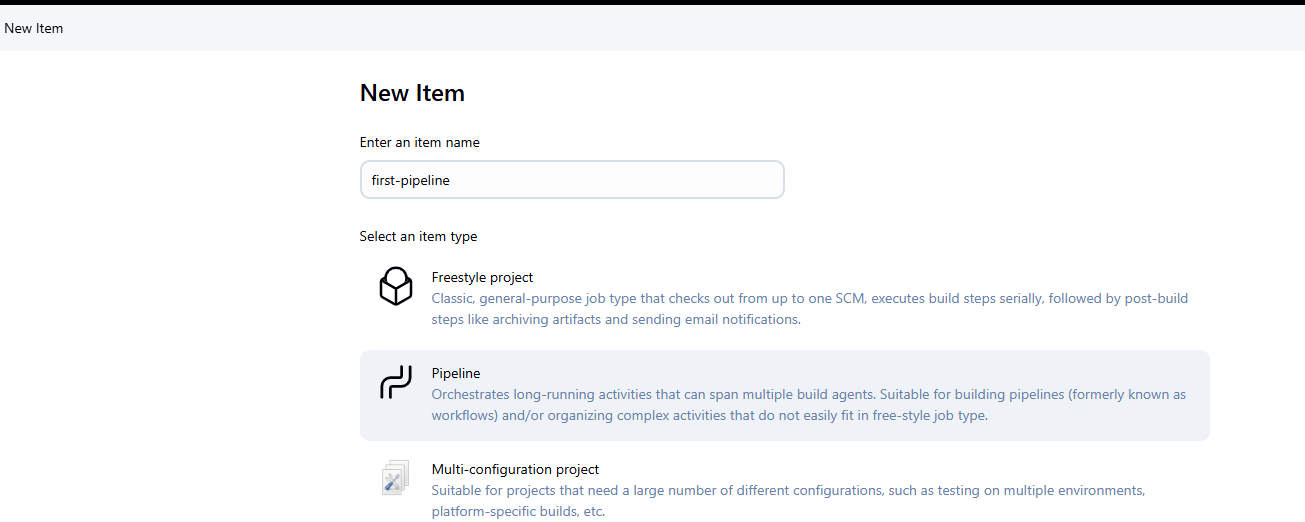
Configure jenkins-slave1



Update Host

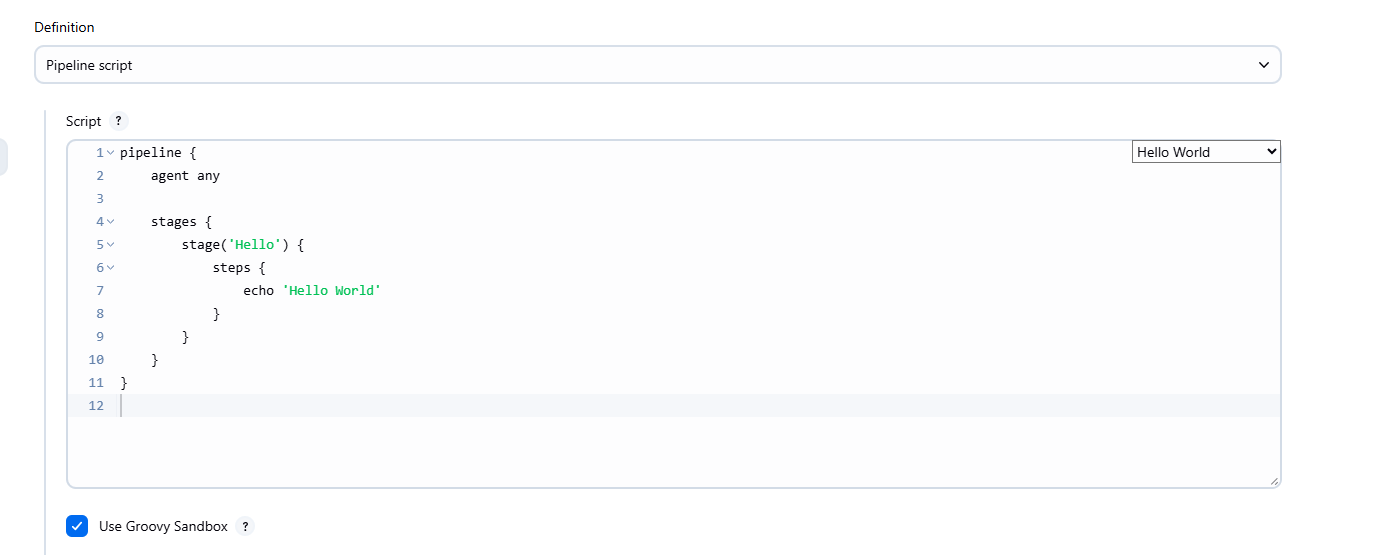
Apply and Save

New Item +

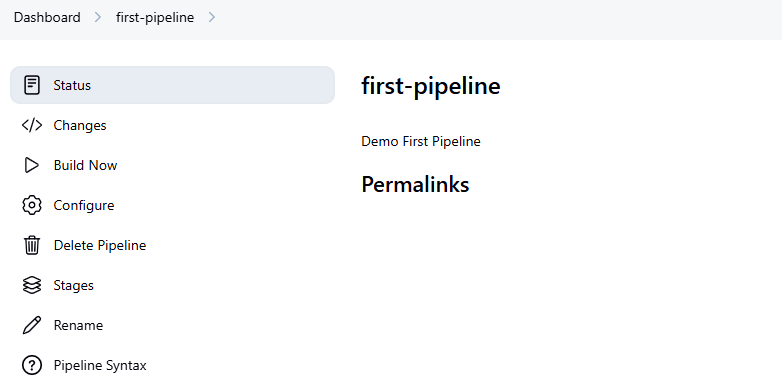


Select Pipeline , click Ok

Pipeline script, Hello World



Apply and Save

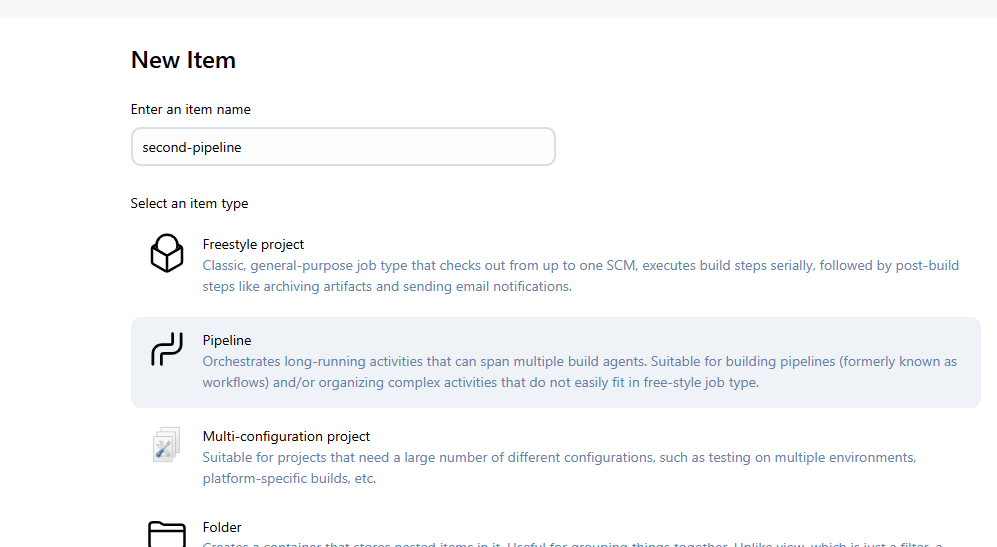


Click Build Now

Console Output

Only one stage is there (Hello World)





pipeline {

agent any

stages {

stage('git clone') {

steps {

echo 'Cloning Git repo'

}

}

stage('maven build') {

steps {

echo 'Project build with Maven'

}

}

stage('deploy') {

steps {

echo 'Deploying App with Tomcat'

}

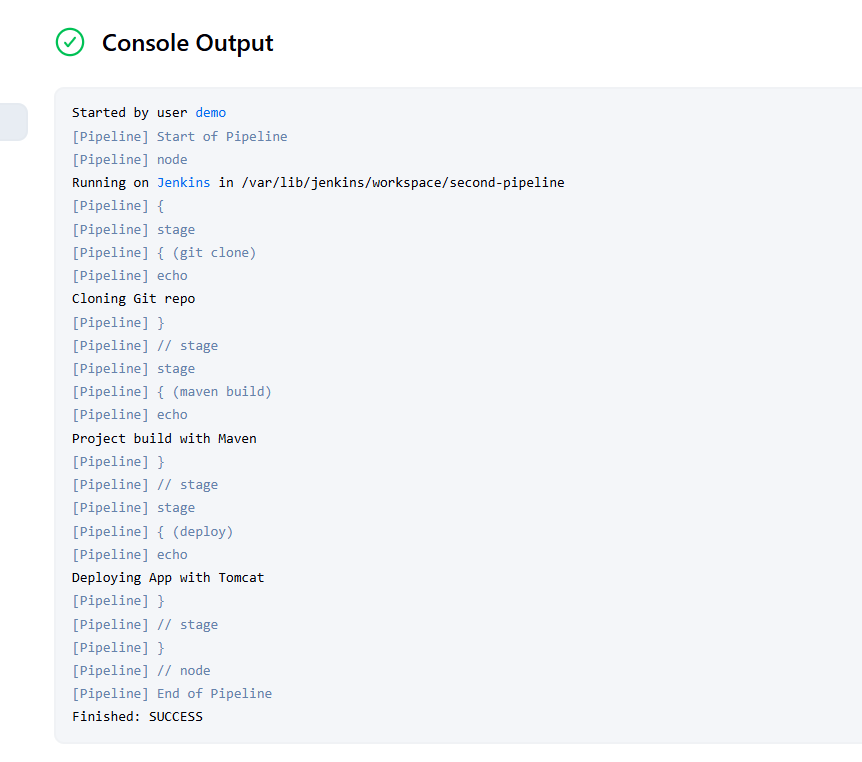
}

}

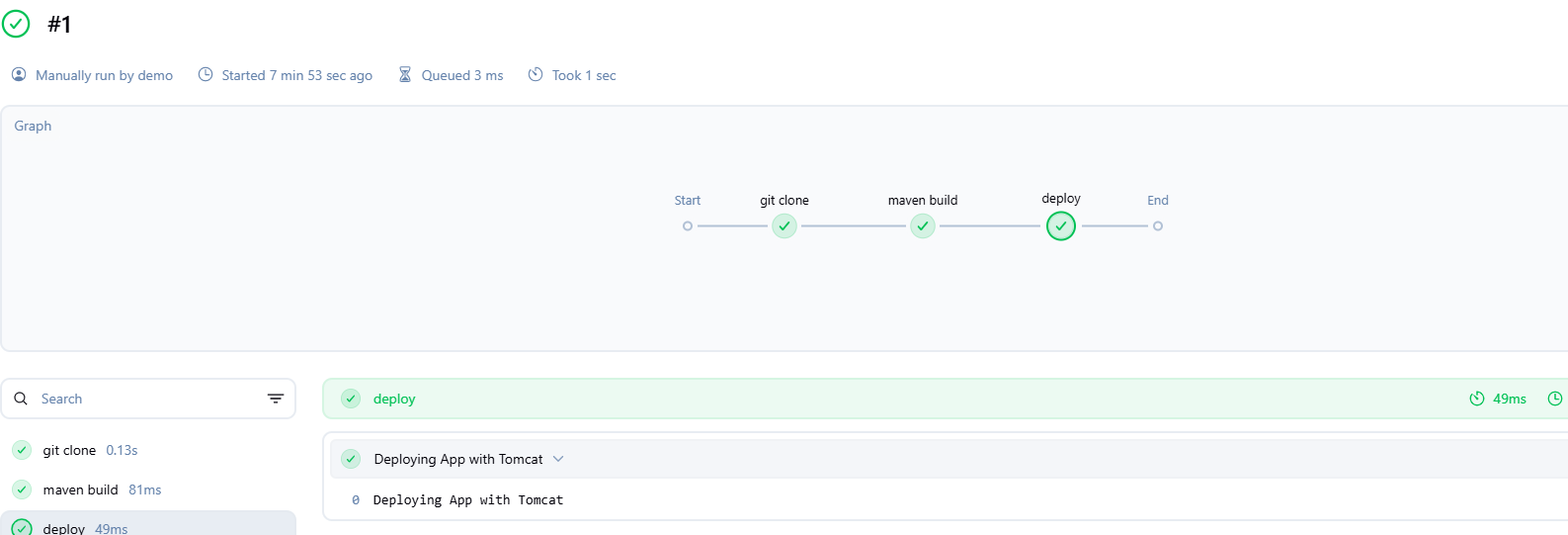
}

Apply and Save

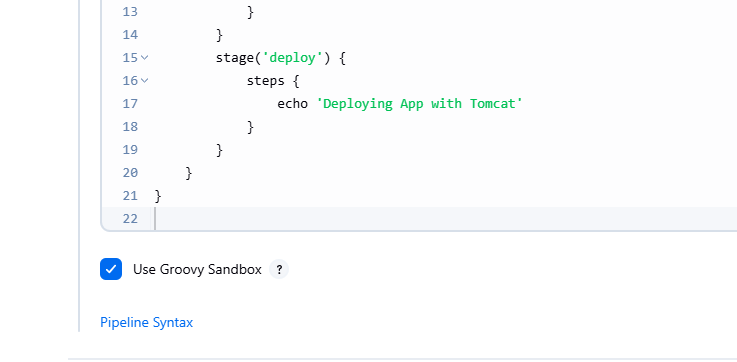
Build Now



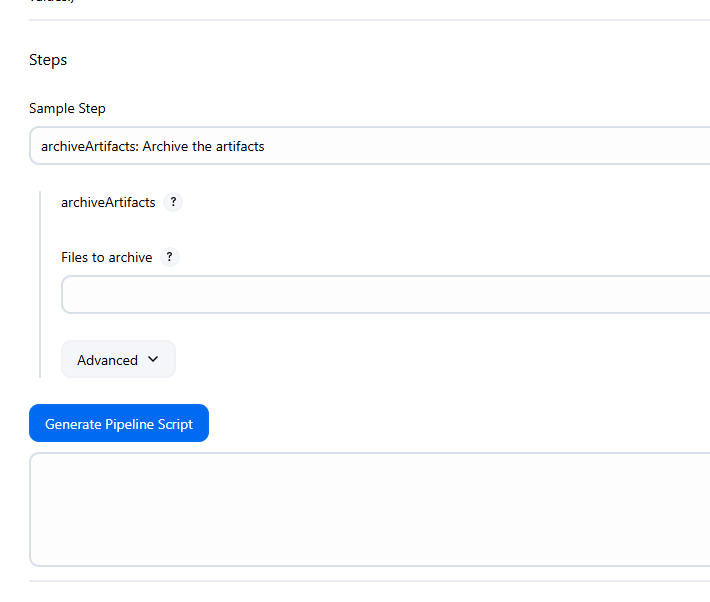
Pipeline Overview

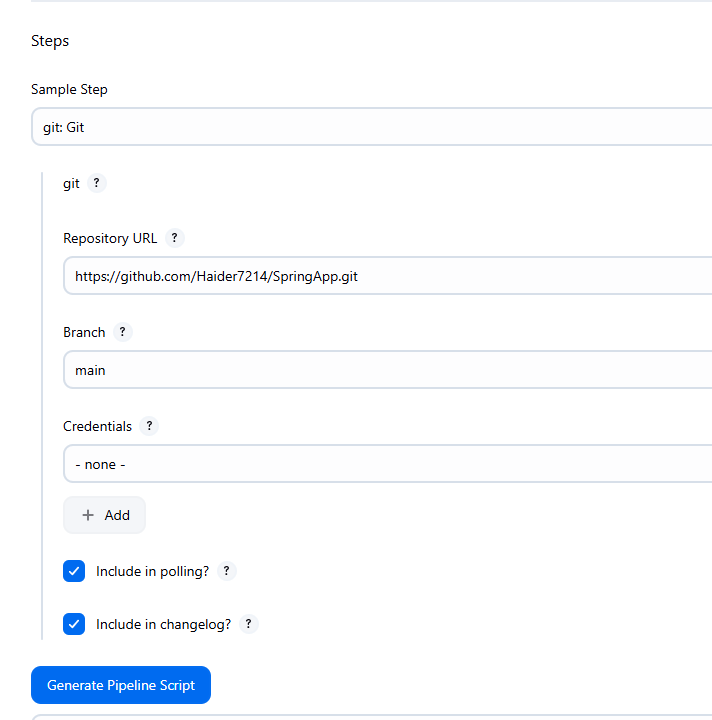


Go back to Configure

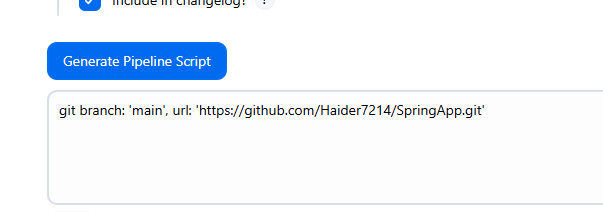


There is something called as Pipeline Syntax





Click Generate Pipeline Script



git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

We need not know the syntax

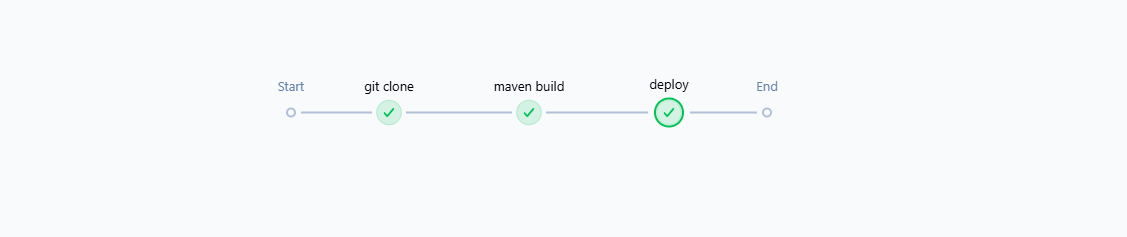
Go back to Pipeline script

Instead of echo add the generated script



Apply and Save

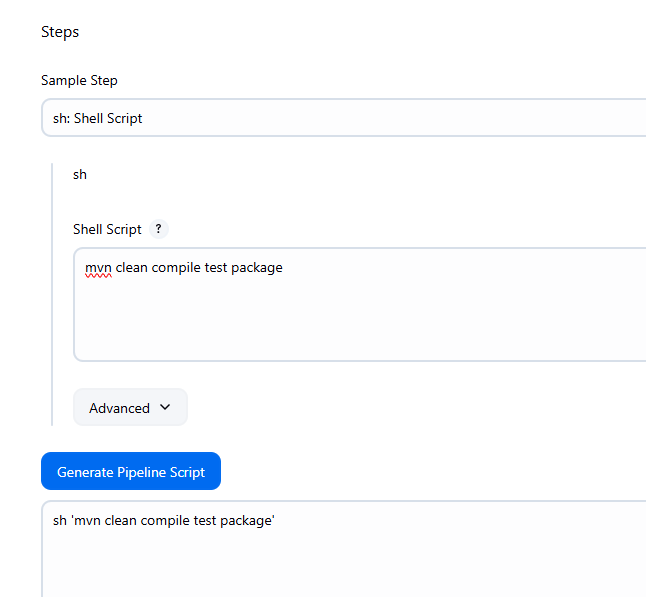
Build Now



It is able to clone the entire project



Shell script --> Generate Pipeline Script



sh 'mvn clean compile test package'

Update Pipeline script



Apply and Save

Build Now

It failed because of mvn: not found



tools {

maven "maven.3.9"

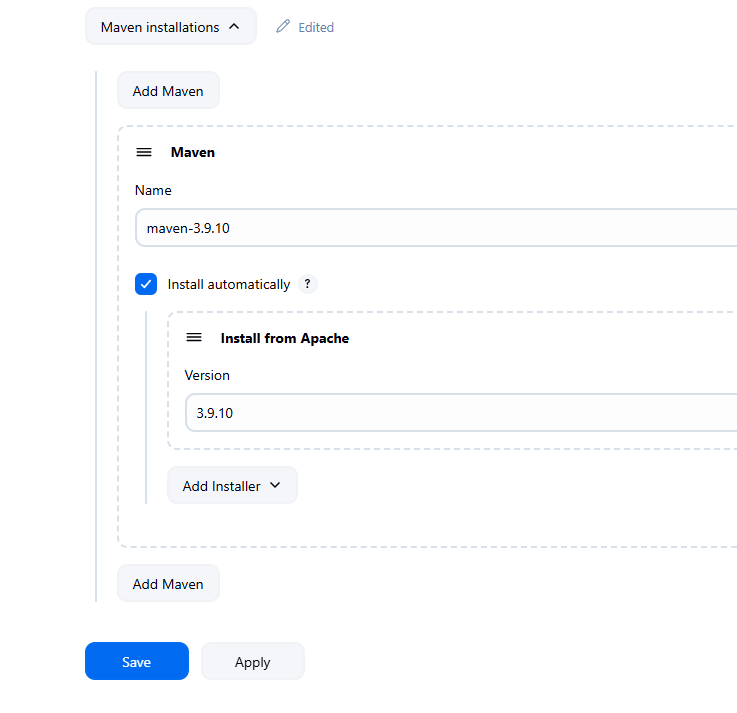
}



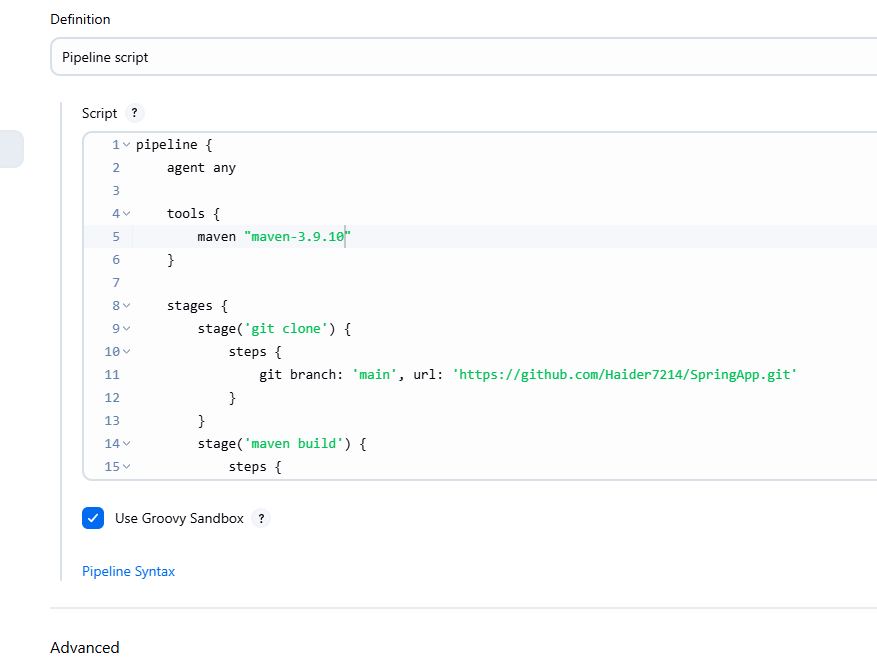
Apply and Save

Manage Jenkins --> Tools

Use the same name here (maven-3.9.10)



Again second-pipeline, Configure

pipeline {

agent any

tools {

maven "maven-3.9.10"

}

stages {

stage('git clone') {

steps {

git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

}

}

stage('maven build') {

steps {

sh 'mvn clean compile test package'

}

}

stage('deploy') {

steps {

echo 'Deploying App with Tomcat'

}

}

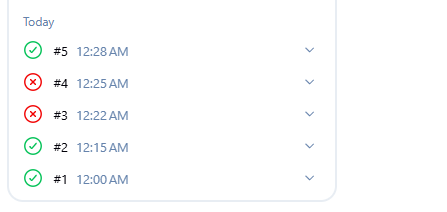
}

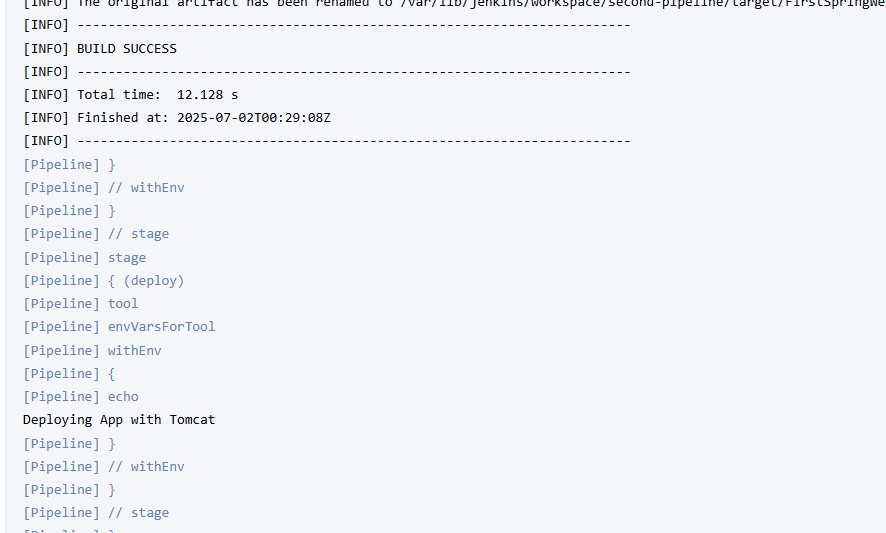
}

Apply and Save

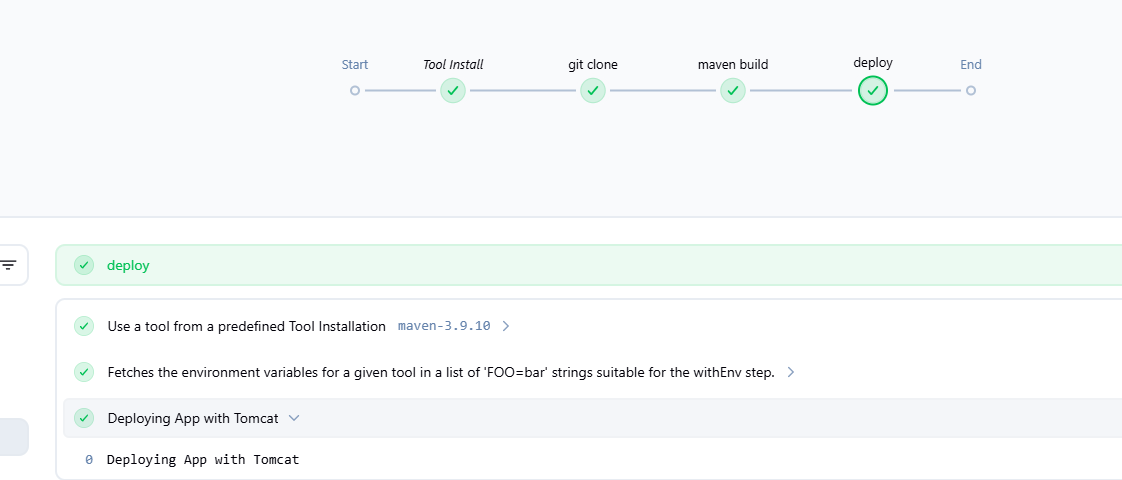
Build Now

Now Build is successful





Now we can see Tool is installed



If one stage is not successful, it will not go to the next stage

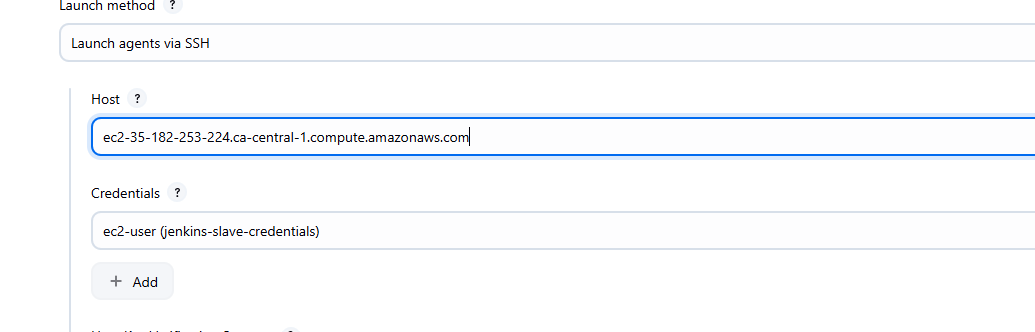
Entire build and deployment process, the code works interlinked with each other

56:00

Restarted Jenkins VM

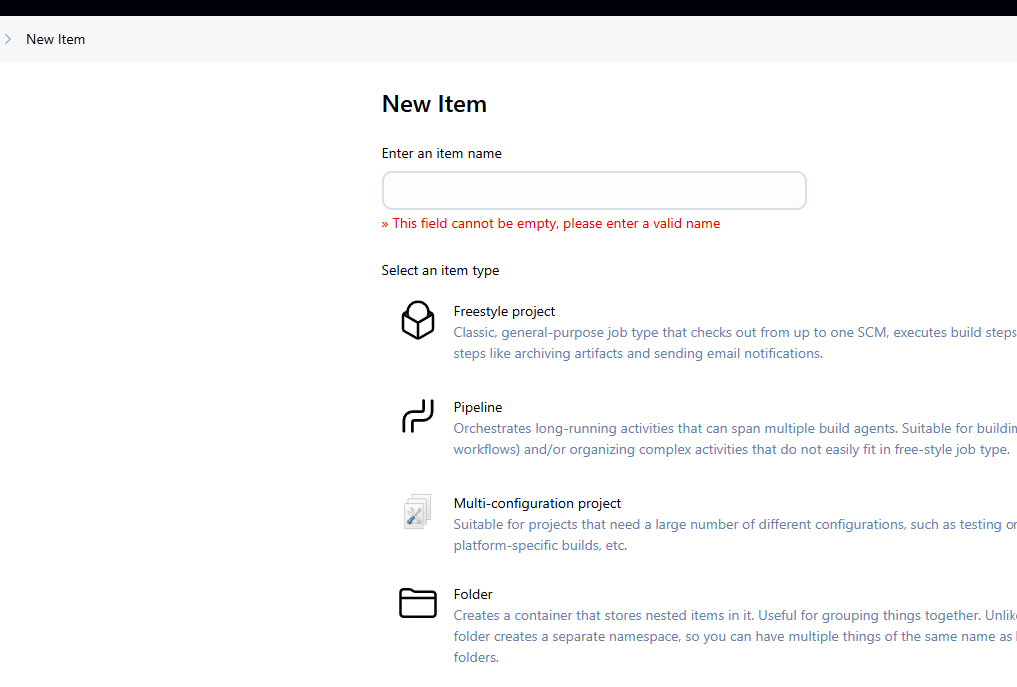
<http://16.52.71.170:8080/manage/configure>

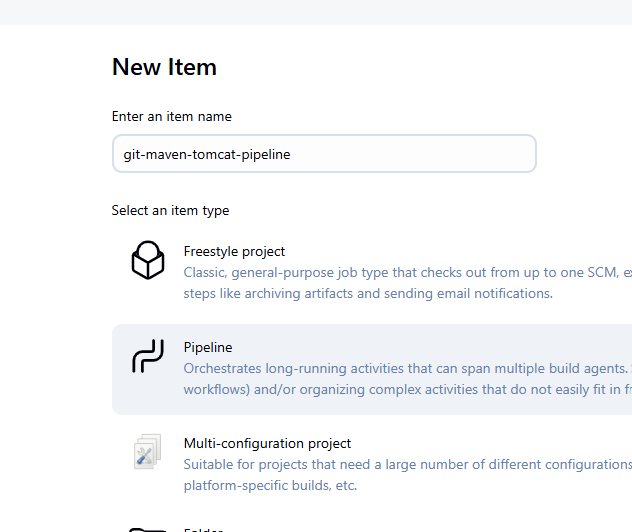
Update slave VM configuration



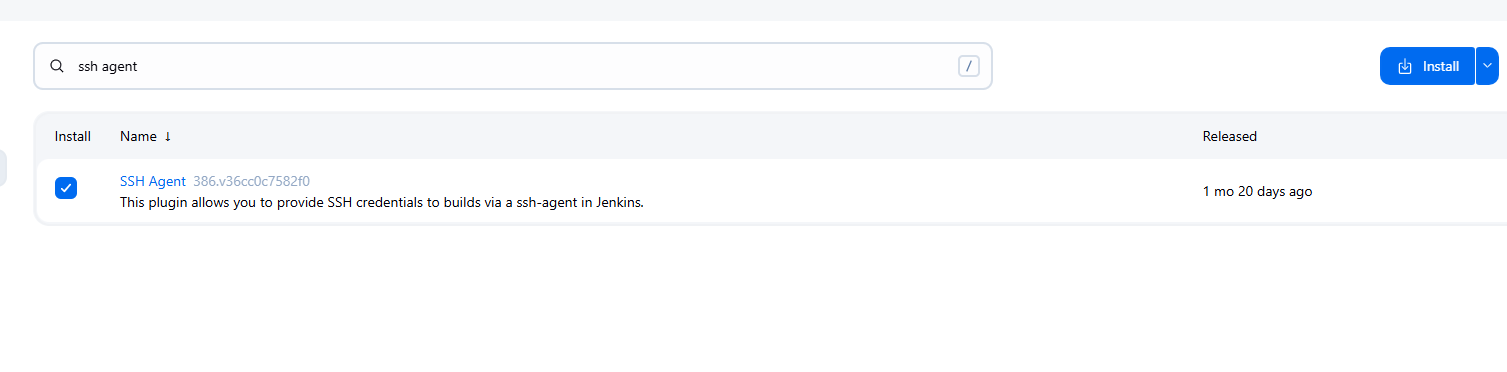
Declarative pipeline with Jenkins + Git + Maven + Tomcat

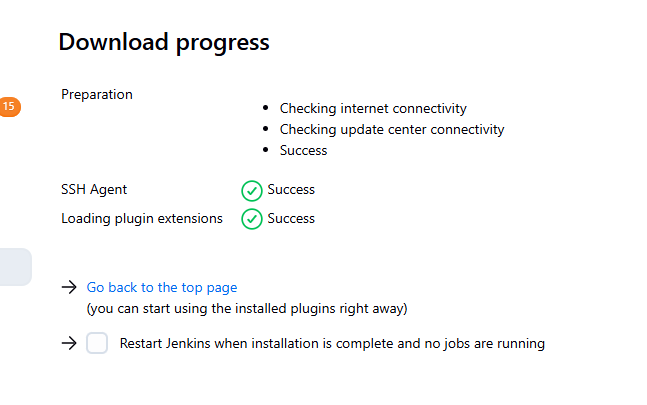
Jenkins Dashboard New Item





Manage Jenkins --> Plugins

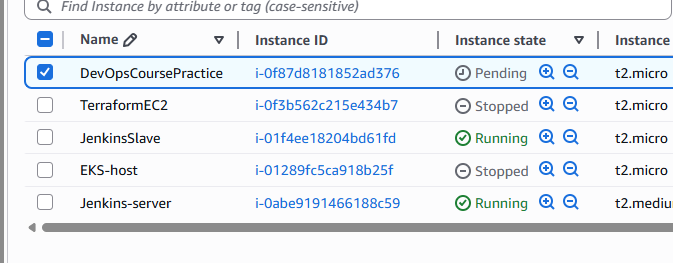






Pipeline Syntax

Turn ON Tomcat server, which I have installed in DevOpsCoursePractice

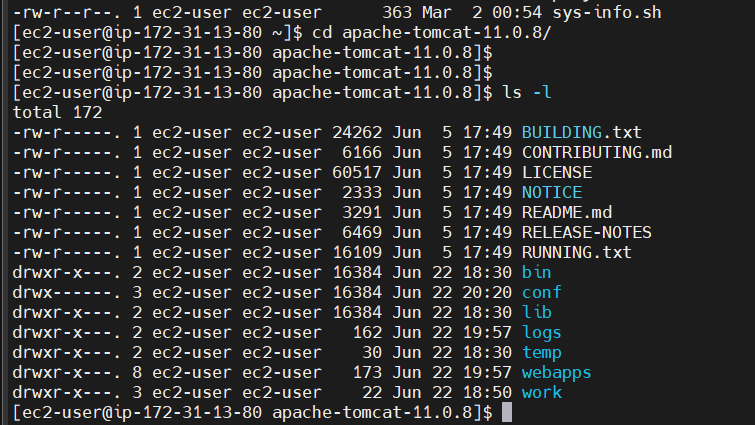


Tomcat Server

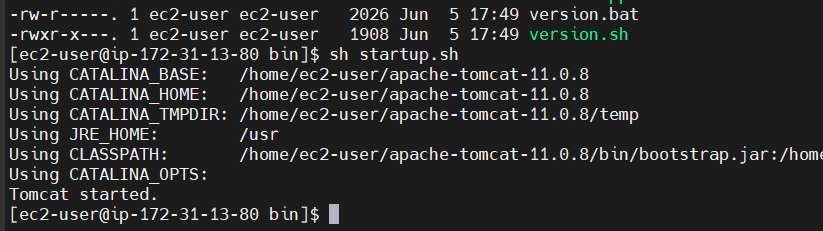
Freestyle plugin -> Deploy to container (lasttime)

Jenkins Server

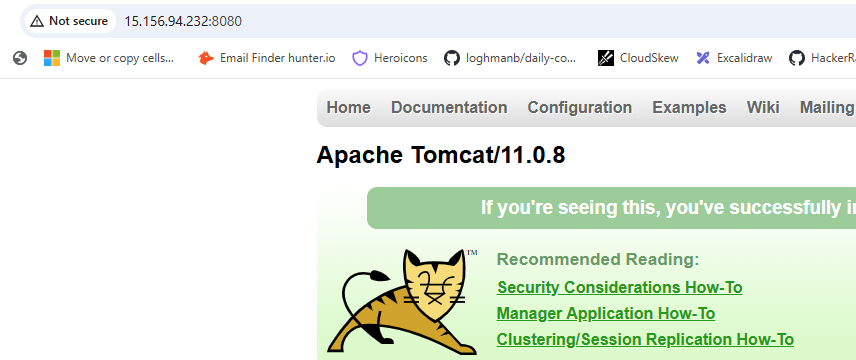
Pipeline plugin -> SSH agent (this time)

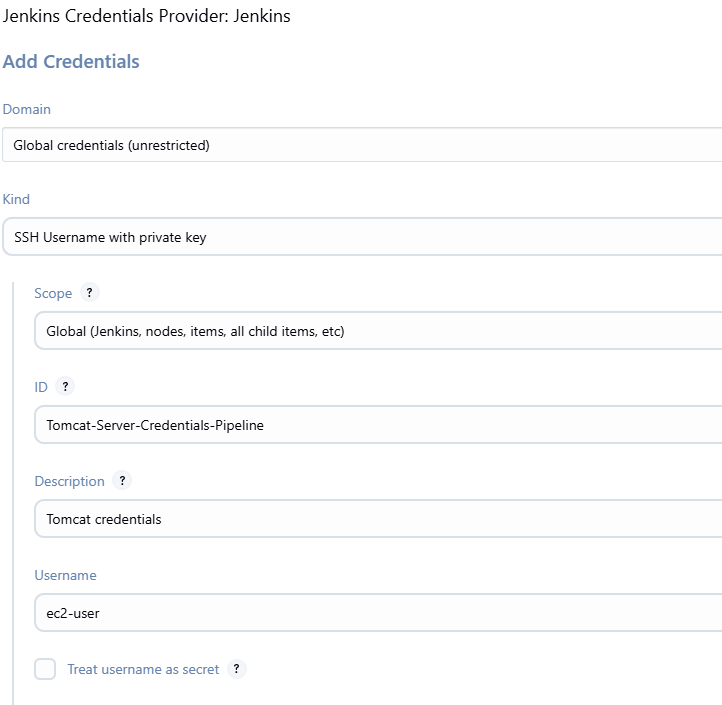


[ec2-user@ip-172-31-13-80 bin]$ sh startup.sh



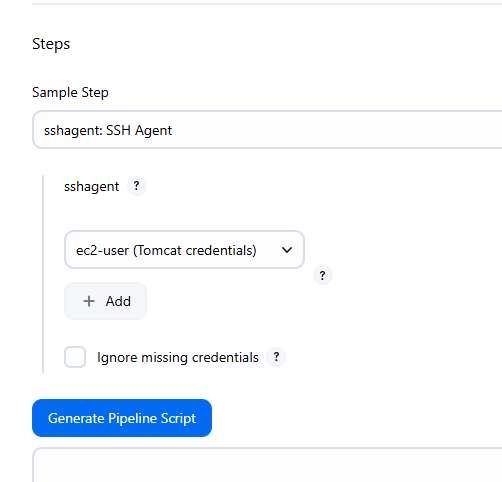
<http://15.156.94.232:8080/>





Copy Paste the Private Key

Click Add

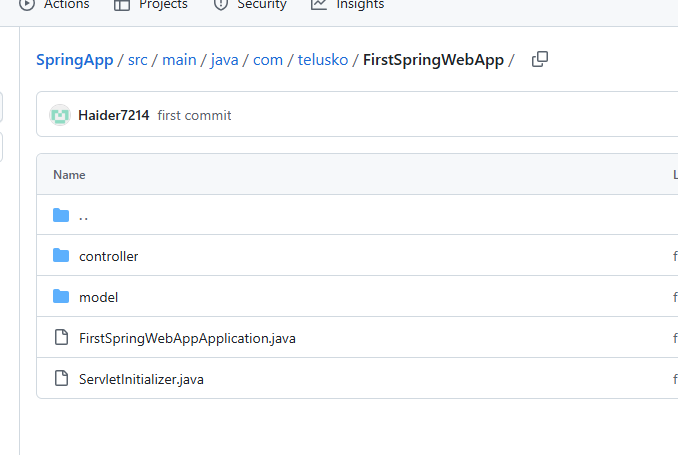


Click Generate Pipeline Script

sshagent(['Tomcat-Server-Credentials-Pipeline']) {

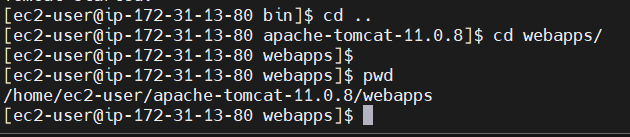
// some block

}



<https://github.com/Haider7214/SpringApp/tree/main/src/main/java/com/telusko/FirstSpringWebApp>

sh 'scp -o StrictHostKeyChecking=no target/FirstSpringWebApp-0.0.1-SNAPSHOT.war'



Copy this path

/home/ec2-user/apache-tomcat-11.0.8/webapps

sh 'scp -o StrictHostKeyChecking=no target/FirstSpringWebApp-0.0.1-SNAPSHOT.war [ec2-user@15.156.94.232:/home/ec2-user/apache-tomcat-11.0.8/webapps'](mailto:ec2-user@15.156.94.232:/home/ec2-user/apache-tomcat-11.0.8/webapps')

pipeline {

agent any

tools {

maven "maven-3.9.10"

}

stages {

stage('git clone') {

steps {

git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

}

}

stage('maven build') {

steps {

sh 'mvn clean compile test package'

}

}

stage('App deployment') {

steps {

sshagent(['Tomcat-Server-Credentials-Pipeline']) {

sh 'scp -o StrictHostKeyChecking=no target/FirstSpringWebApp-0.0.1-SNAPSHOT.war ec2-user@15.156.94.232:/home/ec2-user/apache-tomcat-11.0.8/webapps'

}

}

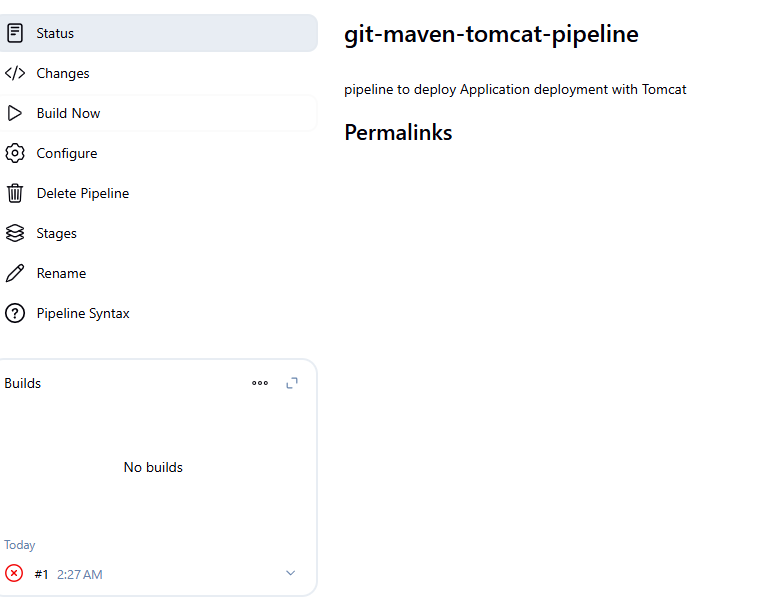
}

}

}

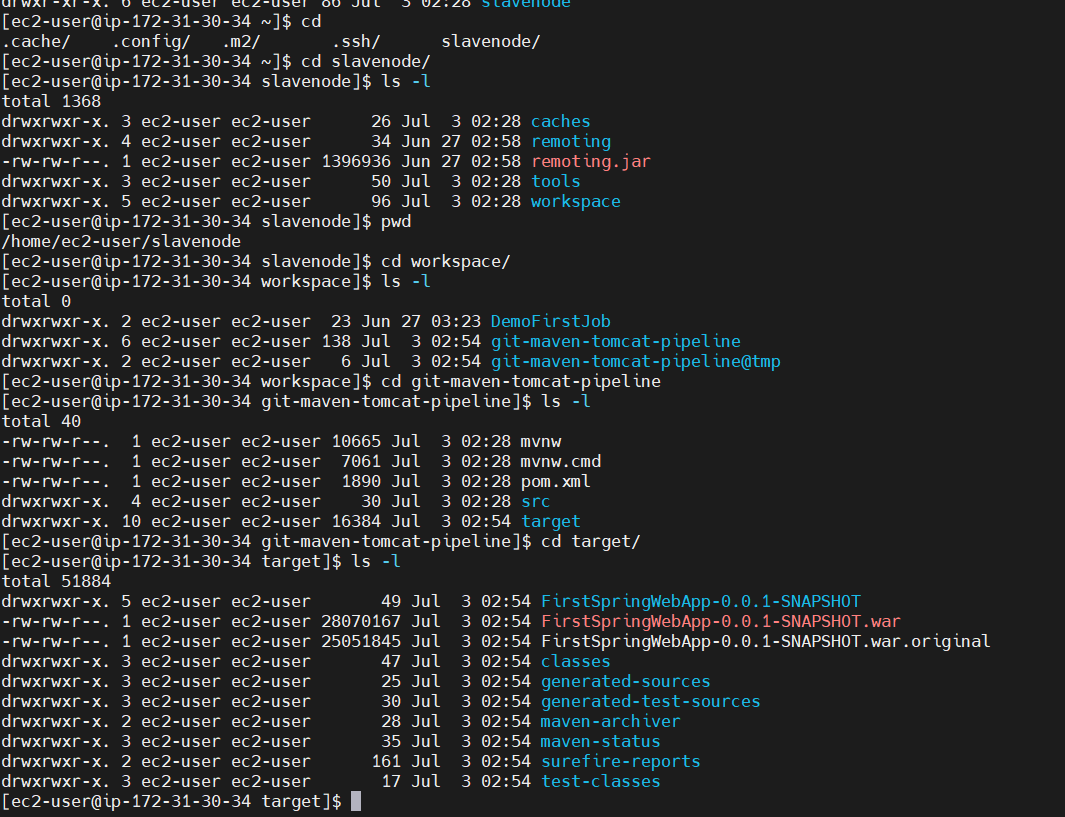
Apply and Save

Build Now



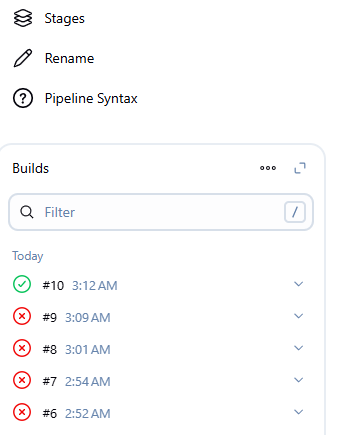


It is there in the slavenode



don’t know why it is not copying into Tomcat-server

I deleted existing FirstSpringWebApp-0.0.1-SNAPSHOT.war



pipeline {

agent any

tools {

maven "maven-3.9.10"

}

stages {

stage('git clone') {

steps {

git branch: 'main', url: 'https://github.com/Haider7214/SpringApp.git'

}

}

stage('maven build') {

steps {

sh 'mvn clean compile test package'

}

}

stage('App deployment') {

steps {

sshagent(['Tomcat-Server-Credentials-Pipeline1']) {

sh 'scp -o StrictHostKeyChecking=no target/FirstSpringWebApp-0.0.1-SNAPSHOT.war ec2-user@15.156.94.232:/home/ec2-user/apache-tomcat-11.0.8/webapps'

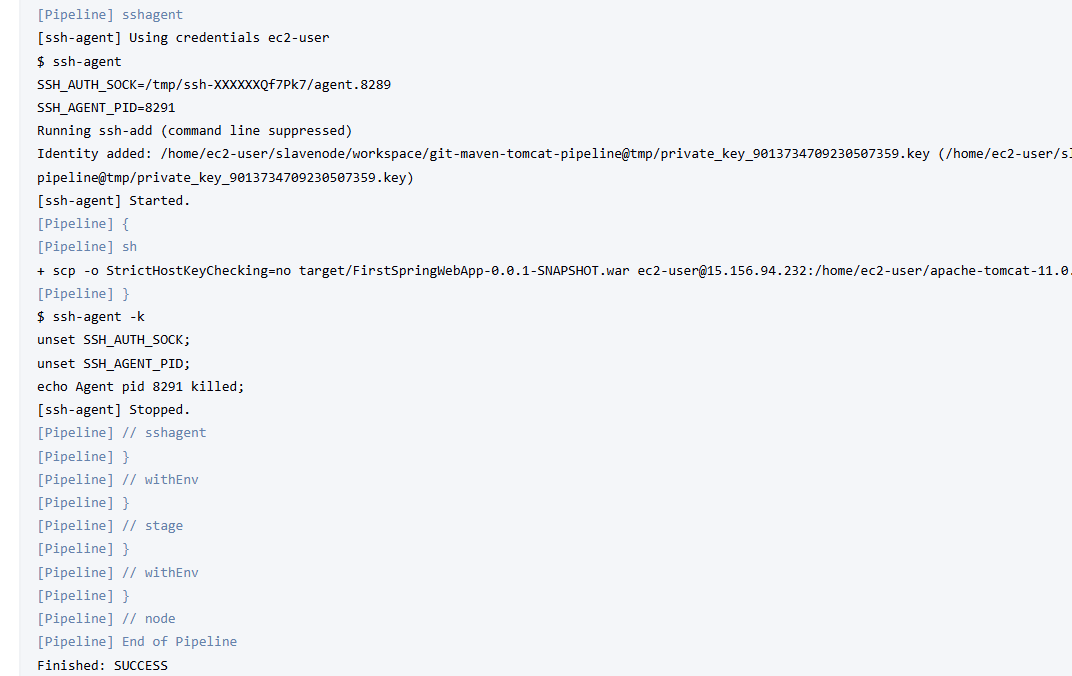
}

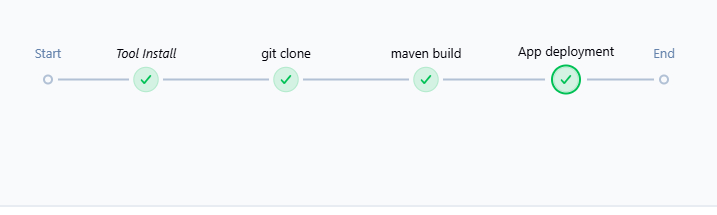
}

}

}

}

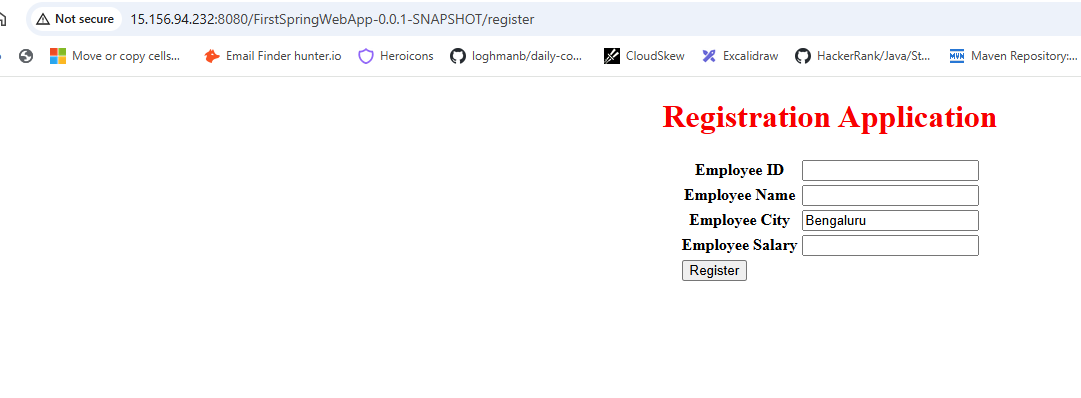




<http://15.156.94.232:8080/manager/html>

Go to Tomcat ---> Manage Apps

<http://15.156.94.232:8080/FirstSpringWebApp-0.0.1-SNAPSHOT/register>



1:30