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

Install SSH Agent Plugin --> Manage Jenkins --> Plugins --> Available Plugins --> SSH Agent --> Install

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'

}

}

}

}

}

Parallel stages:

Some stages I want to execute at the same time. Few stages I want to execute parallely

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('parallel stage') {

parallel {

stage('code-review'){

steps {

echo 'code review'

}

}

stage('nexus-upload'){

steps {

echo 'nexus upload'

}

}

}

}

stage('app deployed') {

steps {

echo 'Deploying App with Tomcat'

}

}

}

}

Shared libraries in Jenkins

Say we want to deploy a microservice application, there are 5 microservices, 5 RestAPIs, maven command will be the same

The code review code will be the same

Instead of writing the same logic again and again, we will reuse the same code

--> Whenever there are multiple pipelines of a project, there might be some common logics in all pipelines. Ex: Artifact upload using Nexus, Code review, Maven build

Instead of we writing this same logics in all our pipelines, we can write this logic at one place and reuse it whenever we need it

--> To achieve this pipeline logic re-usability we can go with the concept of Shared libraries

--> We use Groovy scripting to create shared libraries

def call()

{

sh “mvn clean package”

}

<https://github.com/Haider7214/shared-lib>

<https://github.com/Haider7214/shared-lib.git>

Jenkins Pipeline with Shared library

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 {

mavenBuild()

}

}

stage('parallel stage') {

parallel {

stage('code-review'){

steps {

echo 'code review'

}

}

stage('nexus-upload'){

steps {

echo 'nexus upload'

}

}

}

}

stage('app deployed') {

steps {

echo 'Deploying App with Tomcat'

}

}

}

}

@Library('demo\_shared\_lib')\_

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('demo message') {

steps {

telusko()

}

}

stage('maven build') {

steps {

mavenBuild()

}

}

stage('parallel stage') {

parallel {

stage('code-review'){

steps {

echo 'code review'

}

}

stage('nexus-upload'){

steps {

echo 'nexus upload'

}

}

}

}

stage('app deployed') {

steps {

echo 'Deploying App with Tomcat'

}

}

}

}

mavenBuild.groovy

def call()

{

sh “mvn clean package”

}

1. Scripted pipeline (Groovy)

Multi branch pipeline in Jenkins

SonarQube

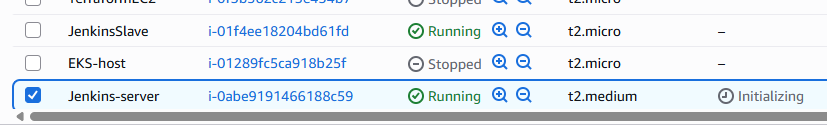
Nexus

CICD pipeline projects

22.00

Declarative pipeline

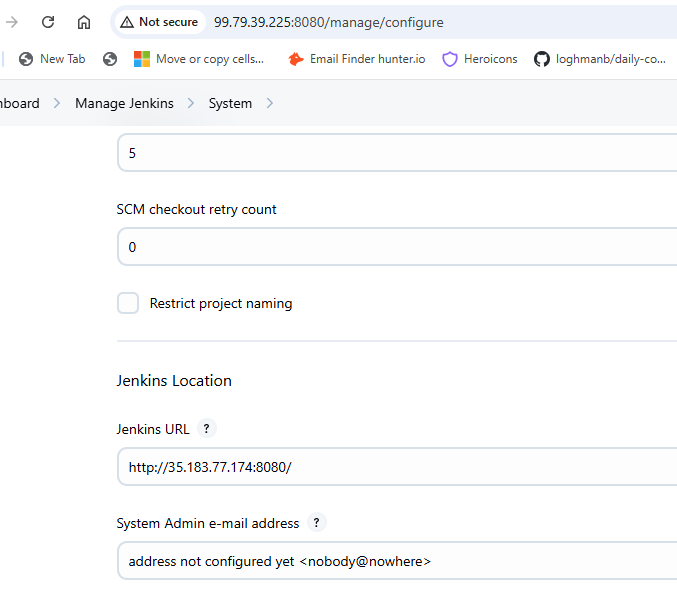
Start JenkinsServer and Slave



Open

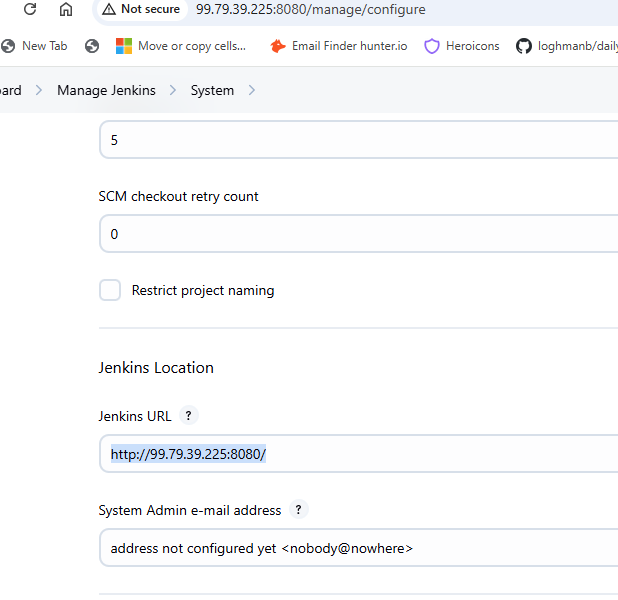
<http://99.79.39.225:8080/manage/configure>

What we have here is different from public IP



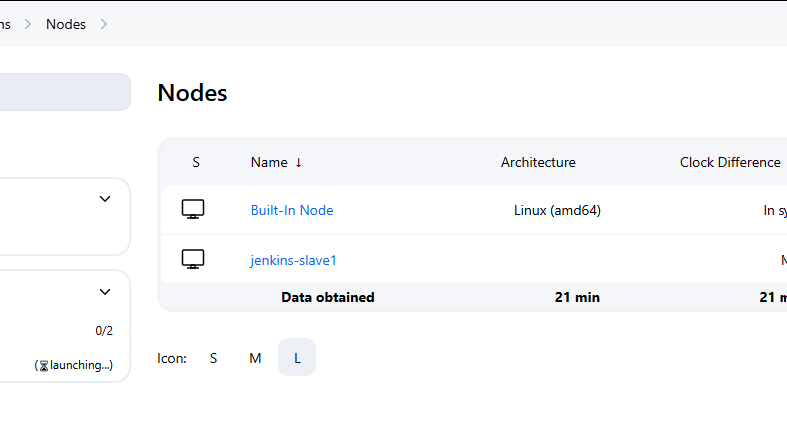
I update Jenkins URL

<http://99.79.39.225:8080/>

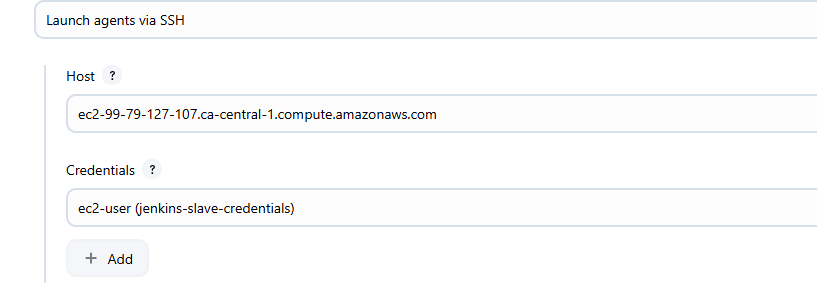


Apply and Save

Go to Nodes



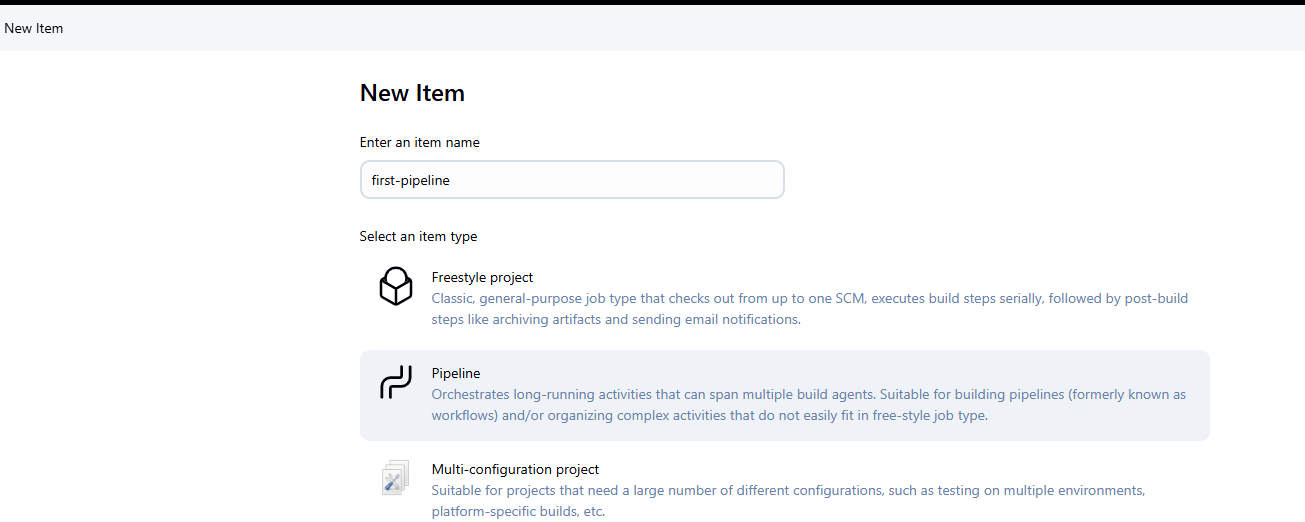
Configure jenkins-slave1



Update Host

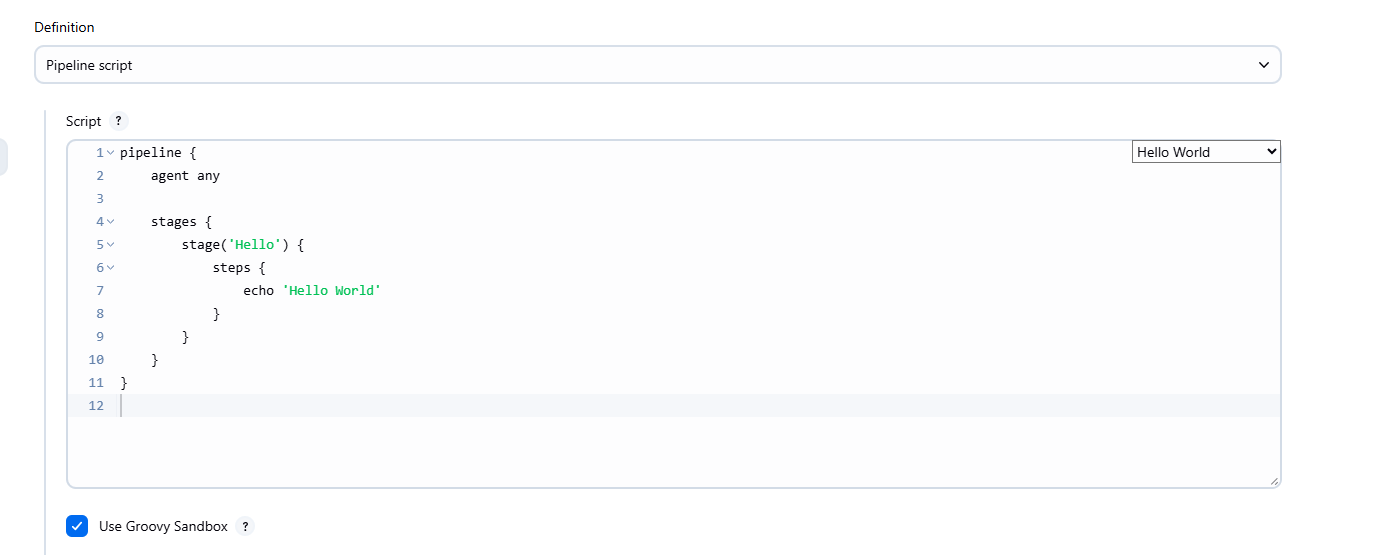
Apply and Save

New Item +

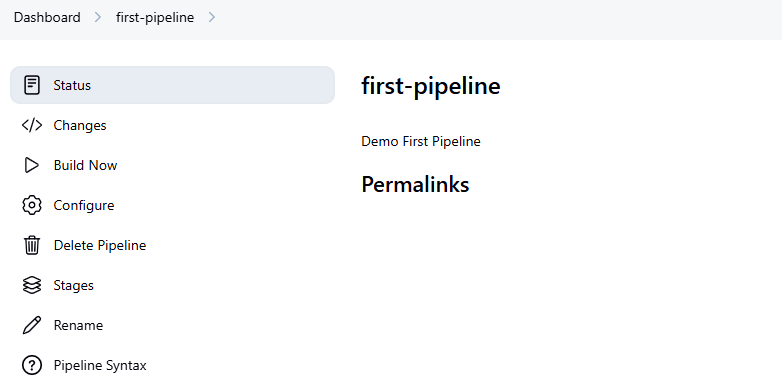


Select Pipeline , click Ok

Pipeline script, Hello World



Apply and Save

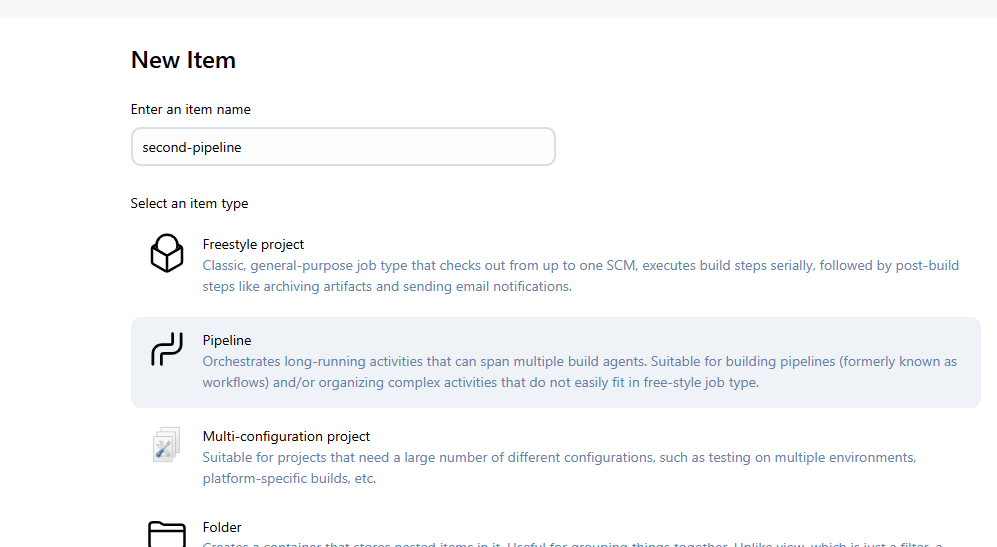


Click Build Now

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'

}

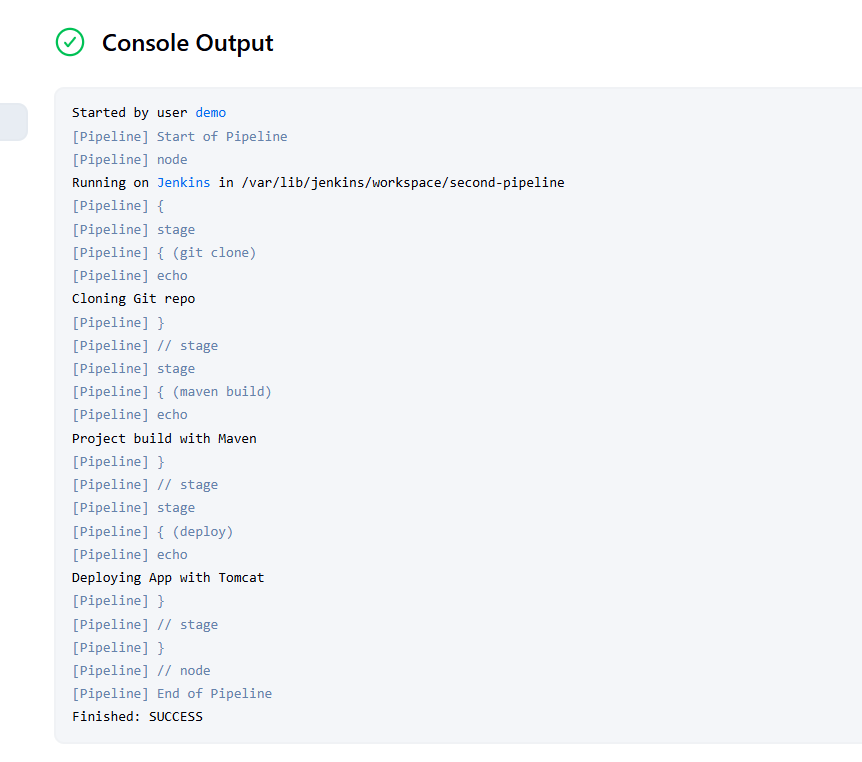
}

}

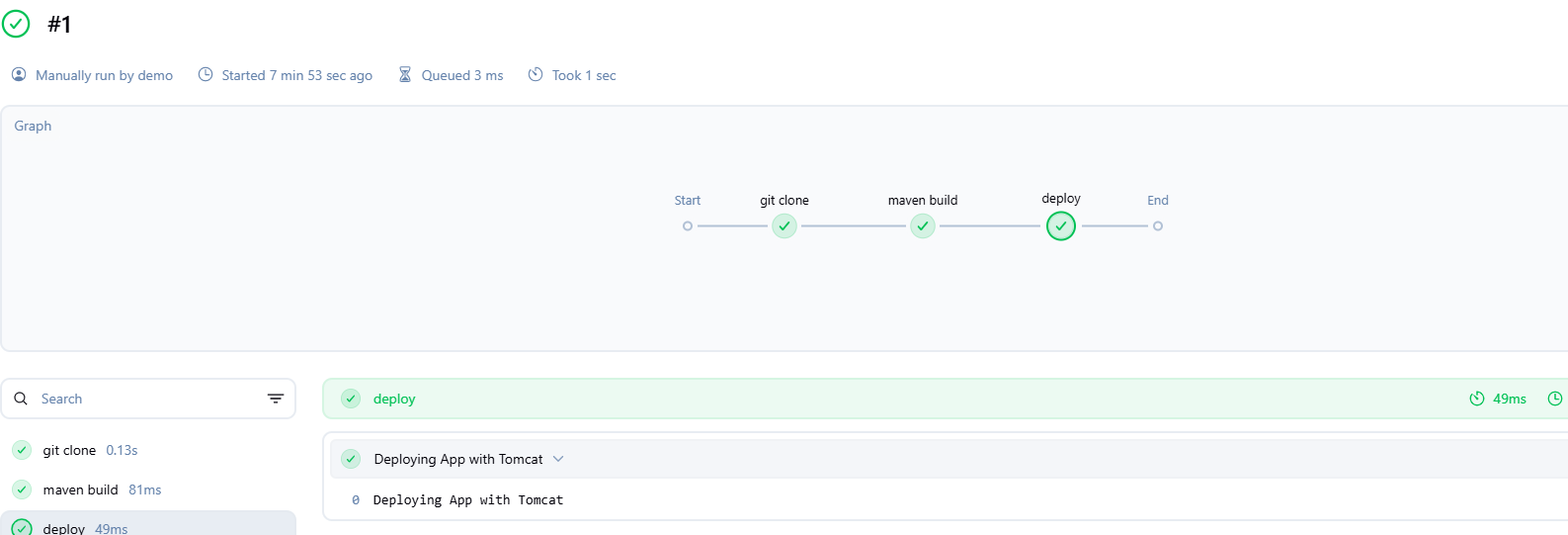
}

Apply and Save

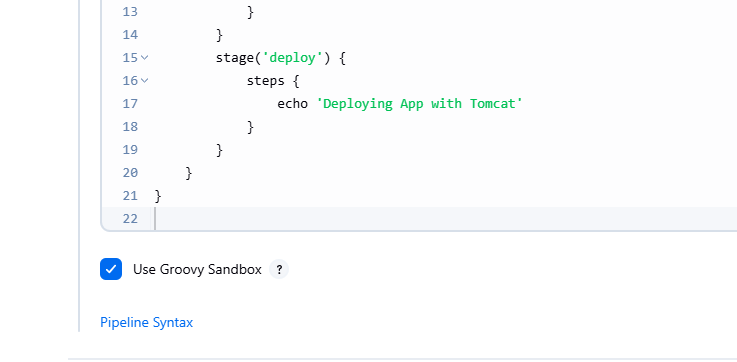
Build Now



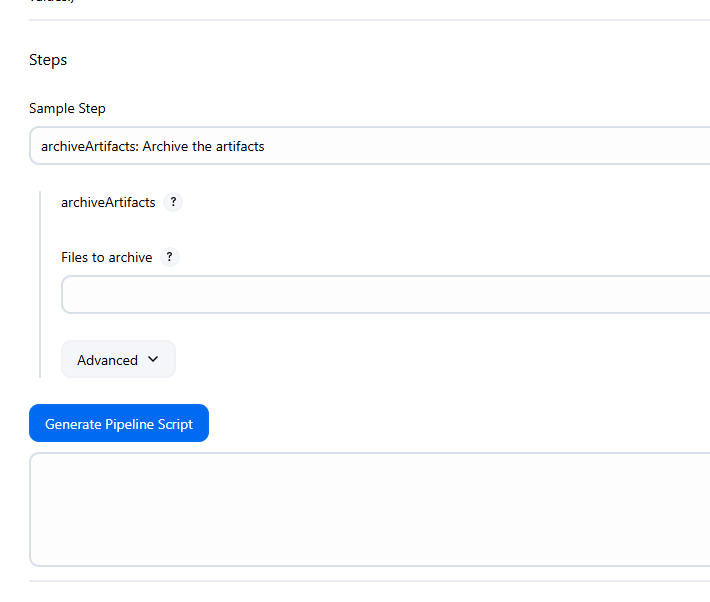
Pipeline Overview

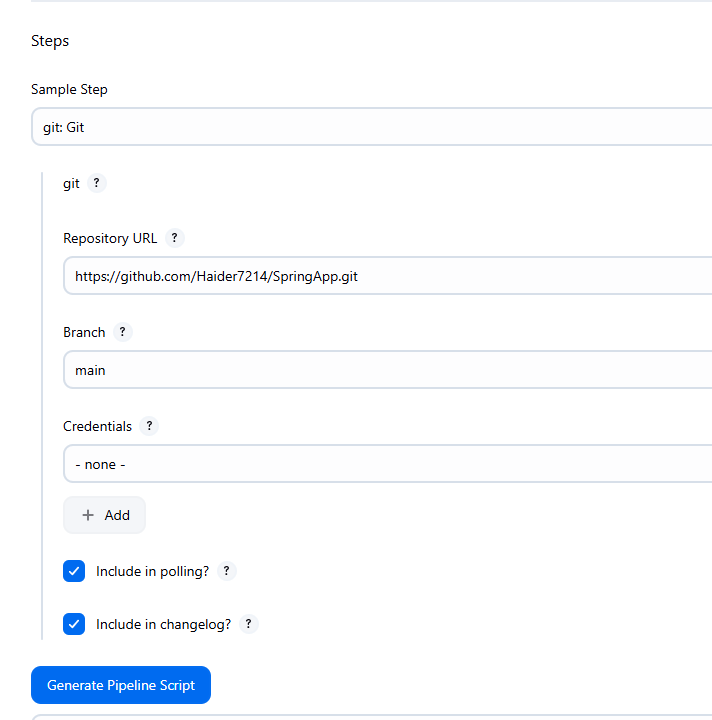


Go back to Configure

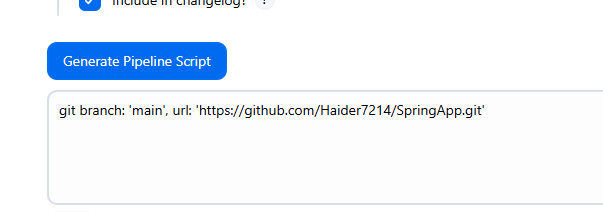


There is something called as Pipeline Syntax





Click Generate Pipeline Script



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

We need not know the syntax

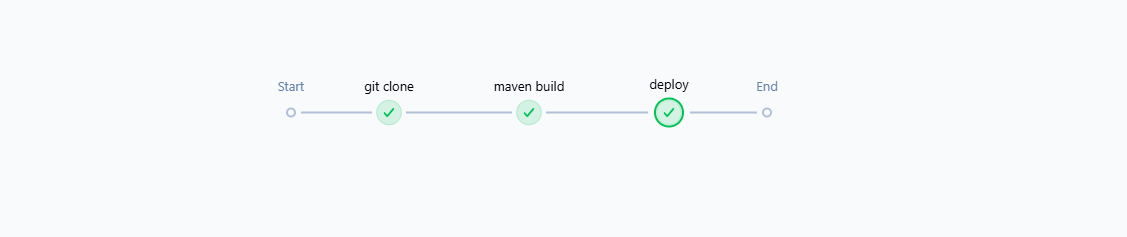
Go back to Pipeline script

Instead of echo add the generated script



Apply and Save

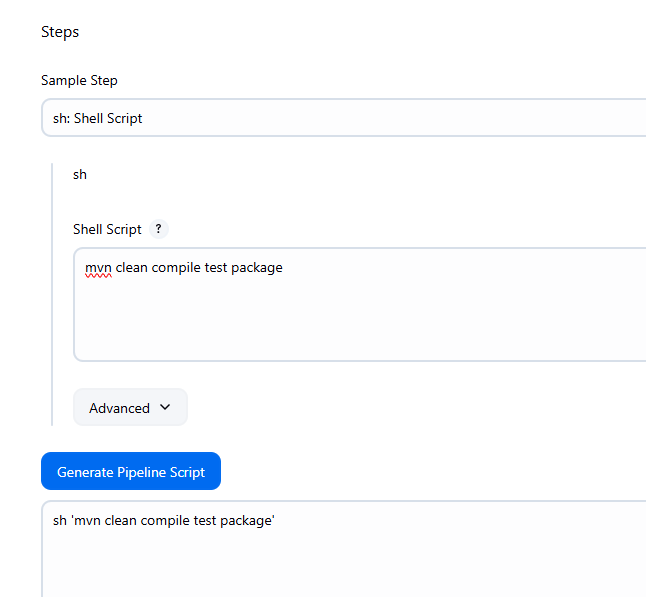
Build Now



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"

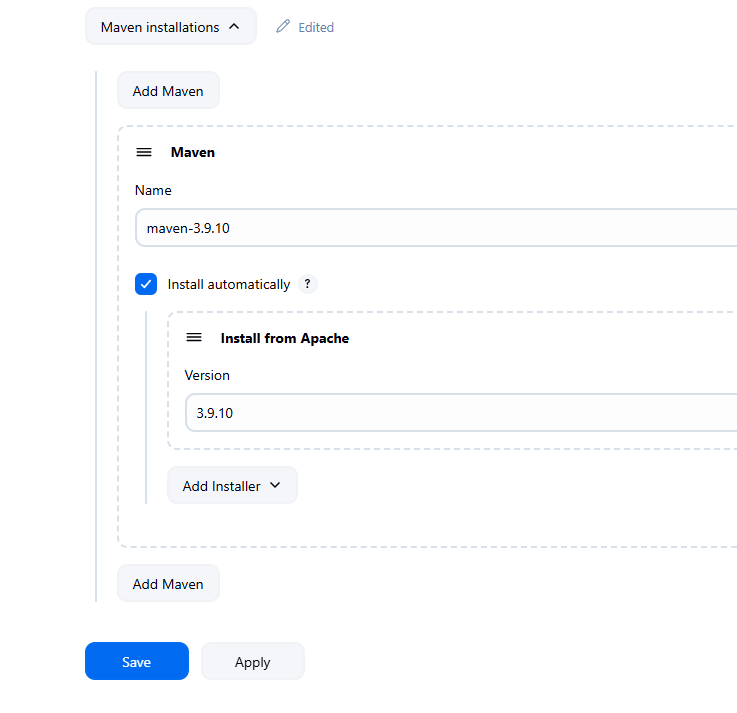
}



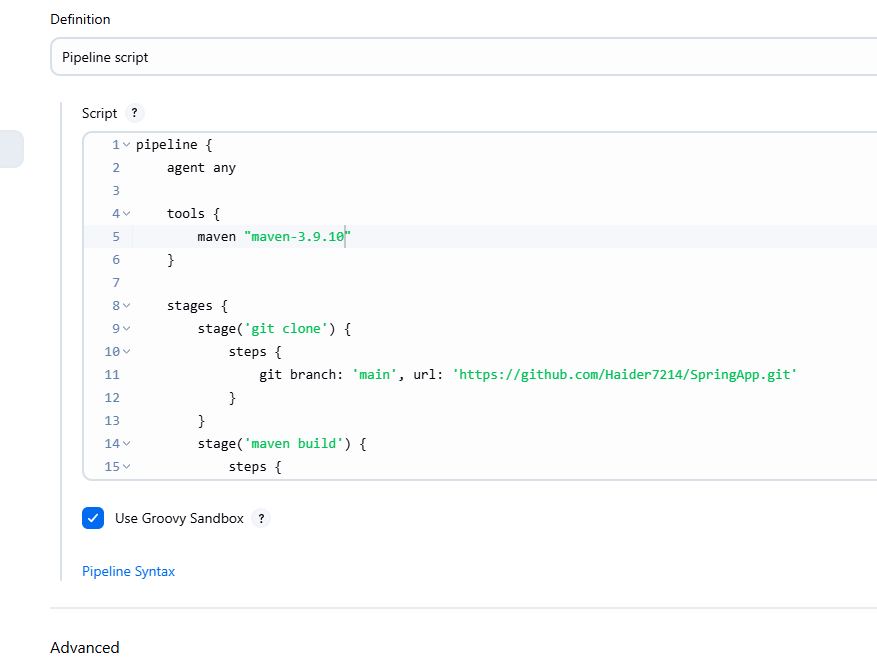
Apply and Save

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'

}

}

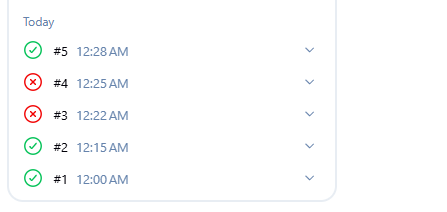
}

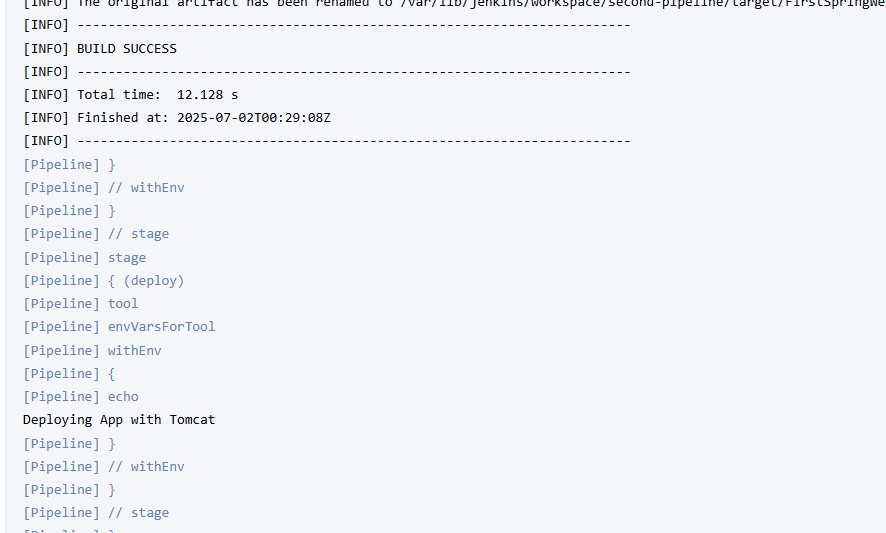
}

Apply and Save

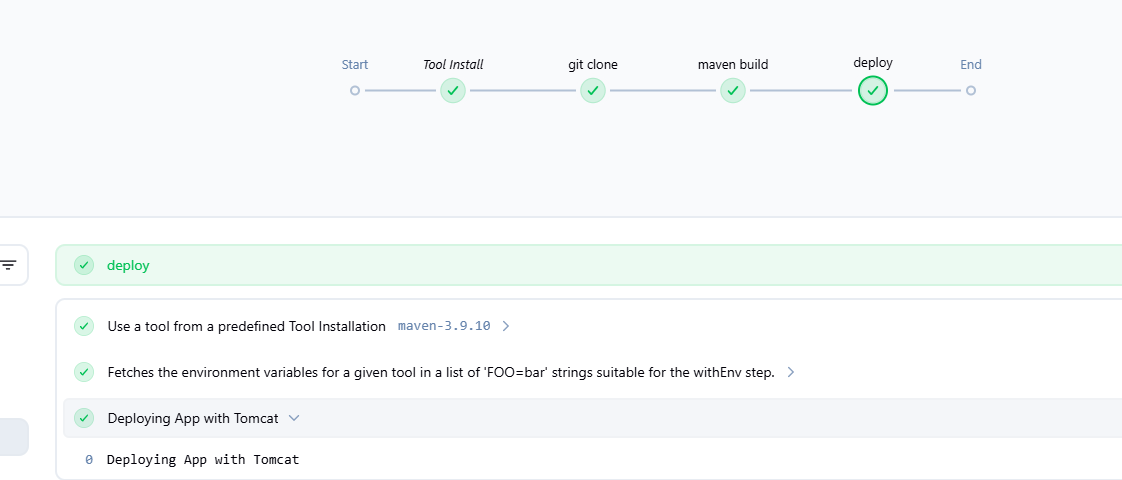
Build Now

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

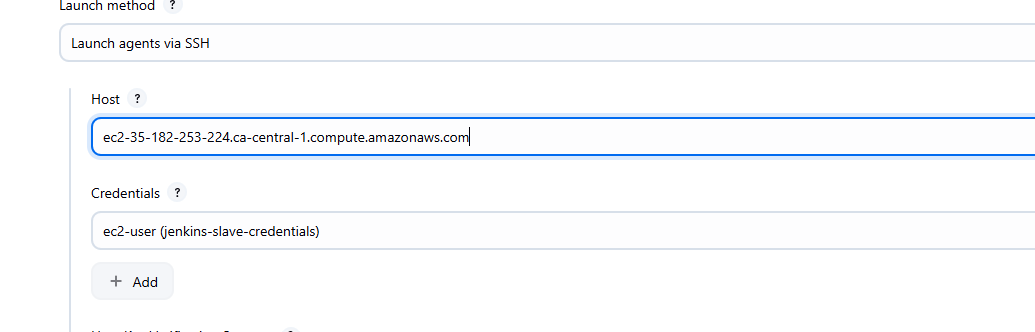
Entire build and deployment process, the code works interlinked with each other

56:00

Restarted Jenkins VM

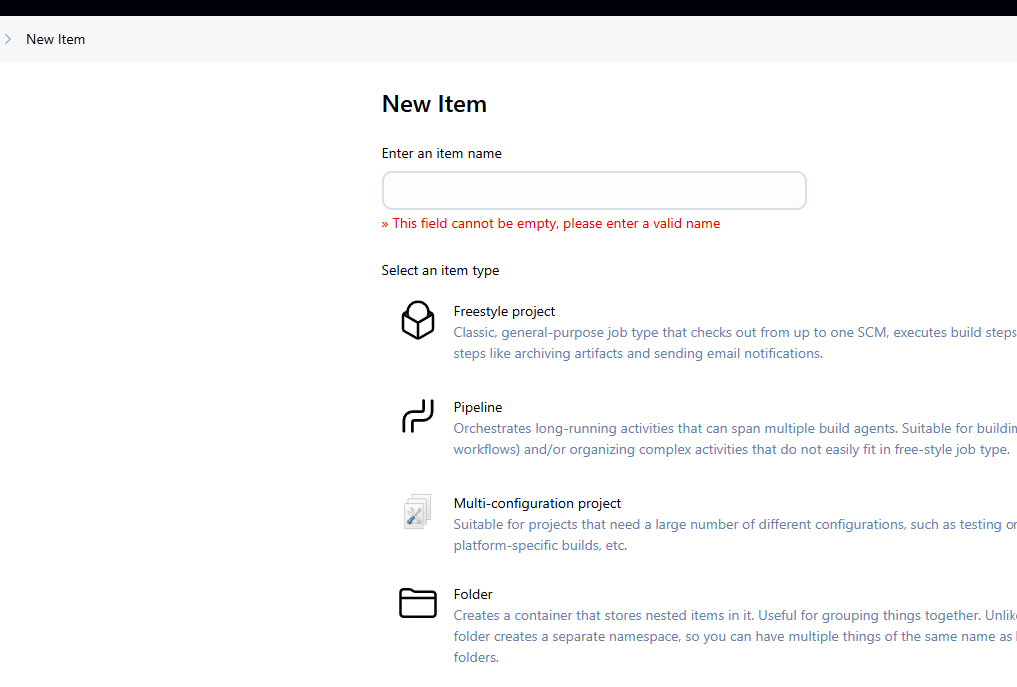
<http://16.52.71.170:8080/manage/configure>

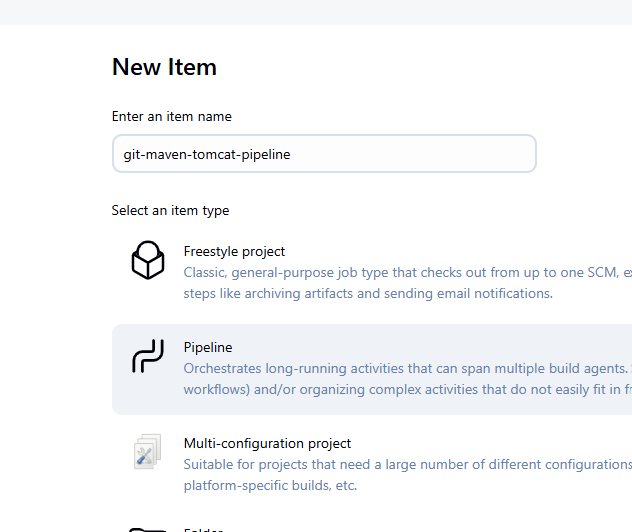
Update slave VM configuration



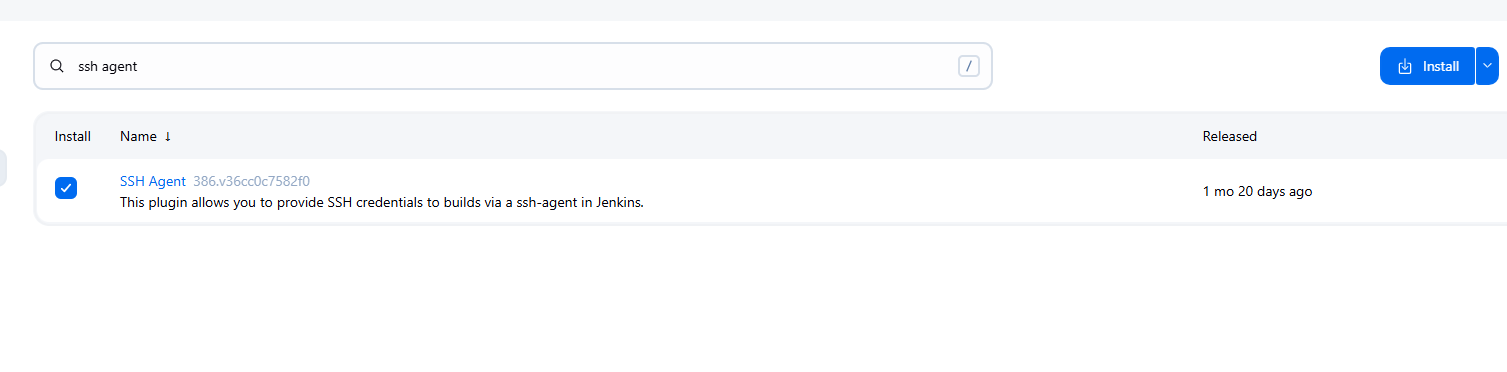
Declarative pipeline with Jenkins + Git + Maven + Tomcat

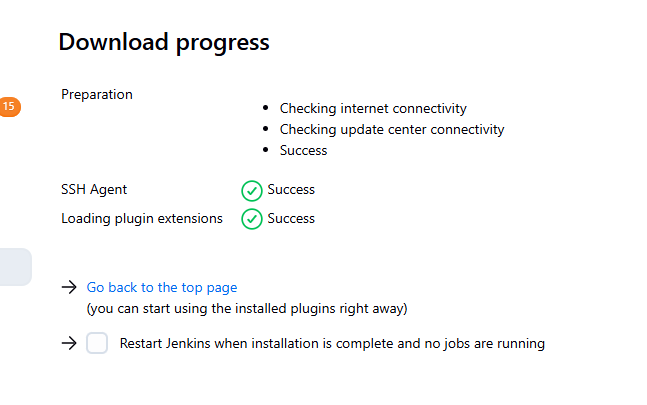
Jenkins Dashboard New Item





Manage Jenkins --> Plugins

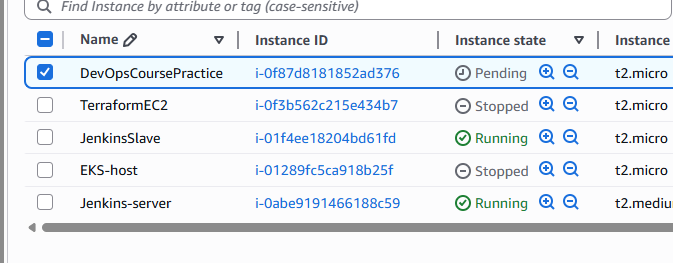






Pipeline Syntax

Turn ON Tomcat server, which I have installed in DevOpsCoursePractice

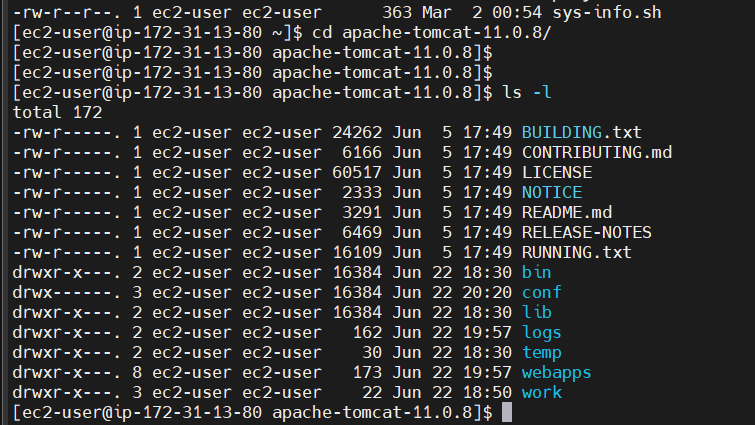


Tomcat Server

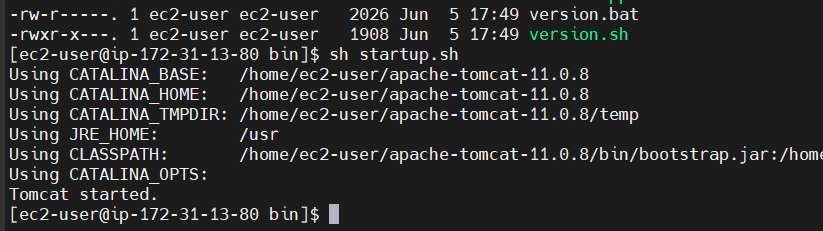
Freestyle plugin -> Deploy to container (lasttime)

Jenkins Server

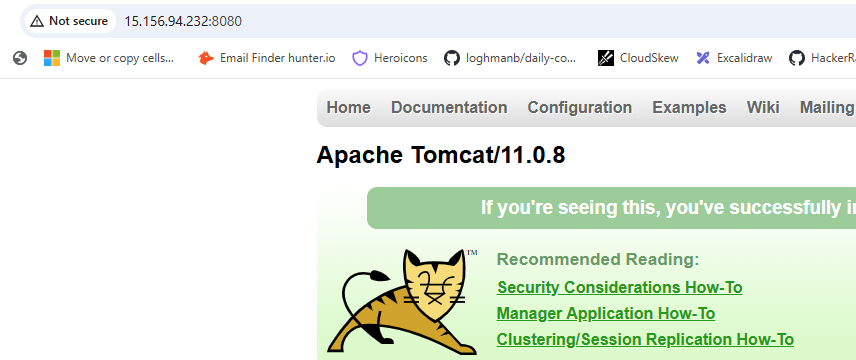
Pipeline plugin -> SSH agent (this time)

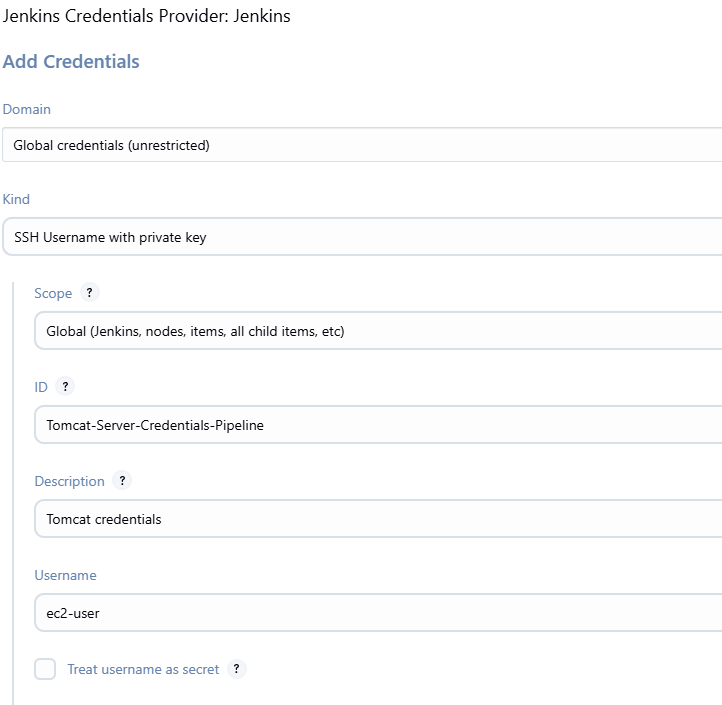


[ec2-user@ip-172-31-13-80 bin]$ sh startup.sh



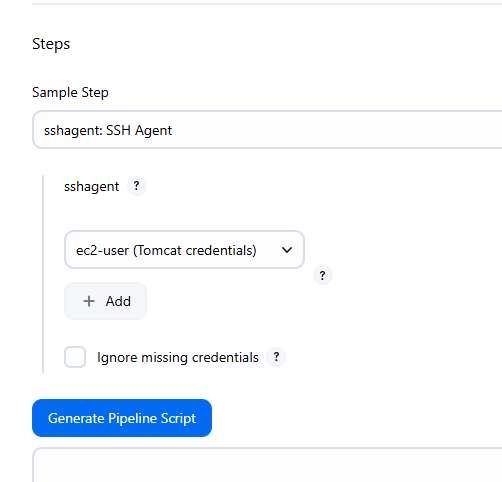
<http://15.156.94.232:8080/>





Copy Paste the Private Key

Click Add

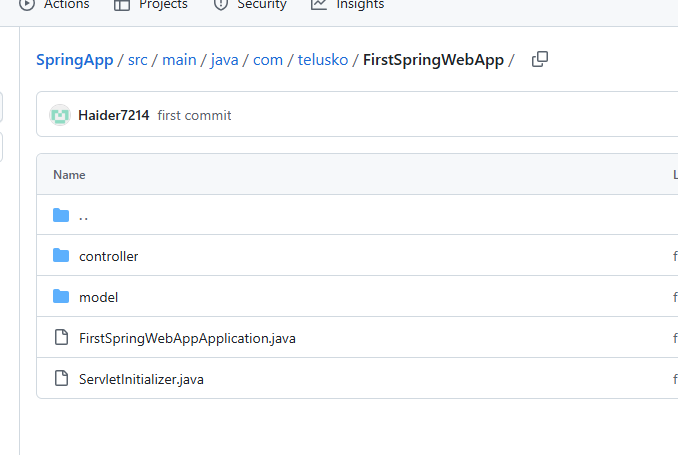


Click Generate Pipeline Script

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

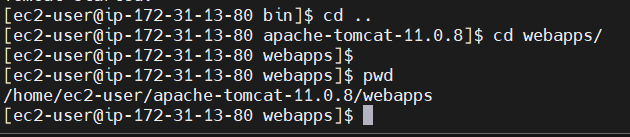
// some block

}



<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'

}

}

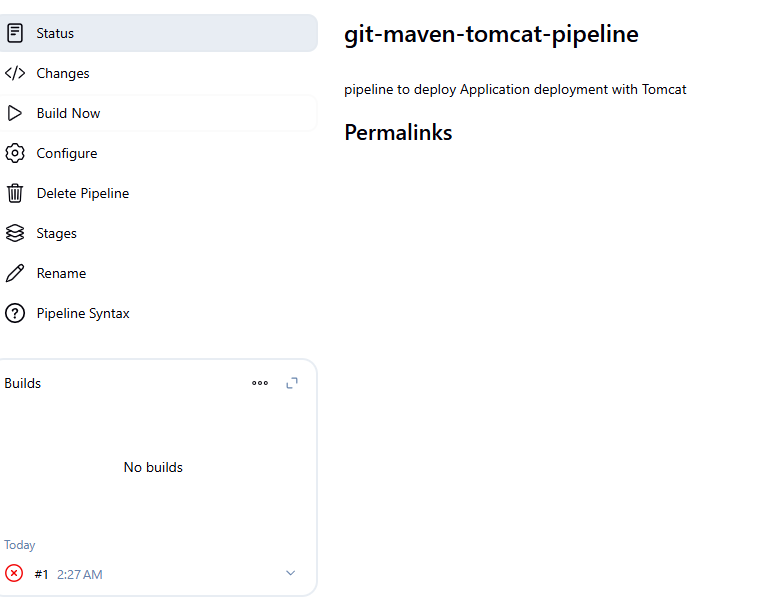
}

}

}

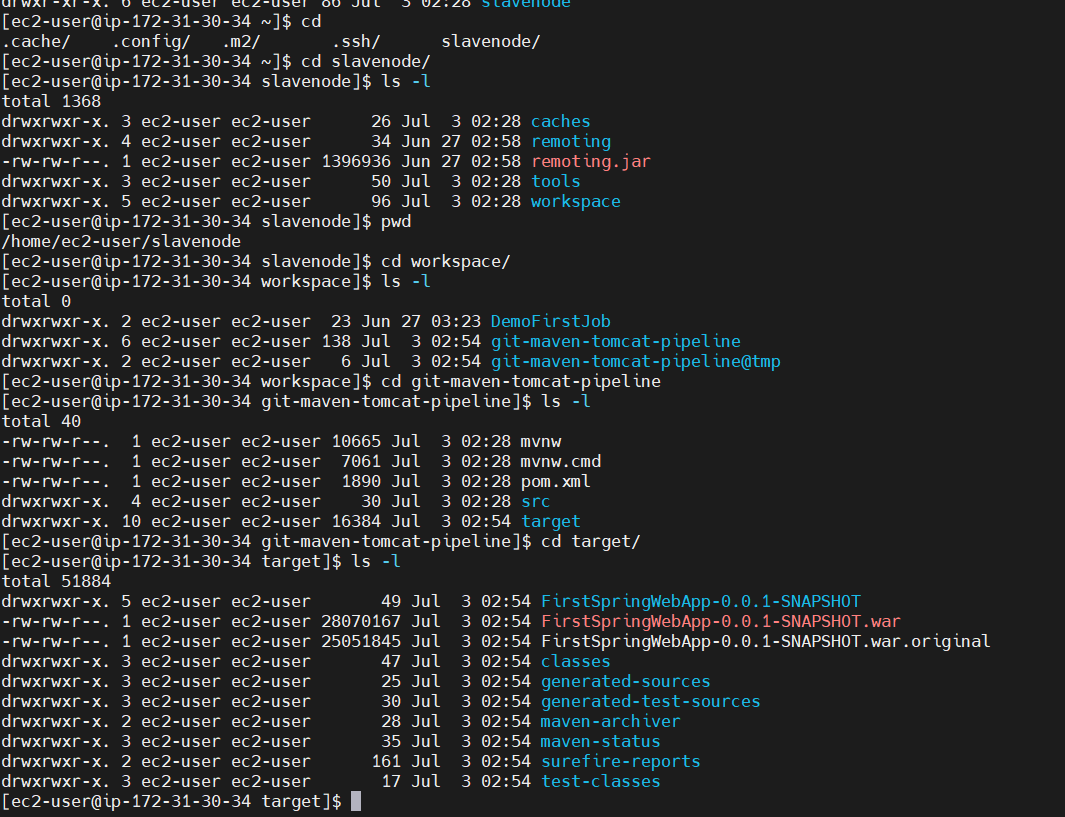
Apply and Save

Build Now



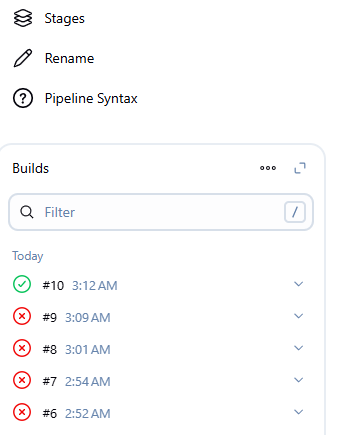


It is there in the slavenode



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'

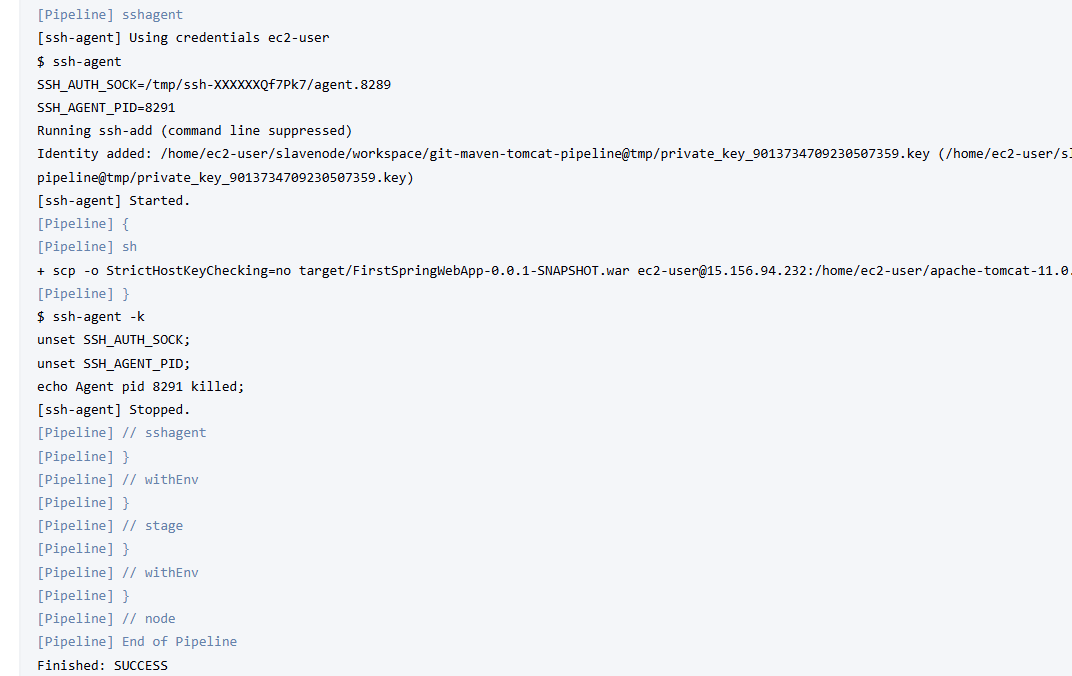
}

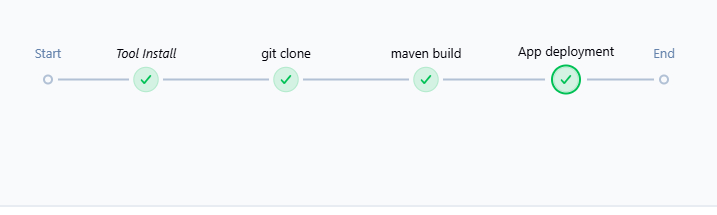
}

}

}

}

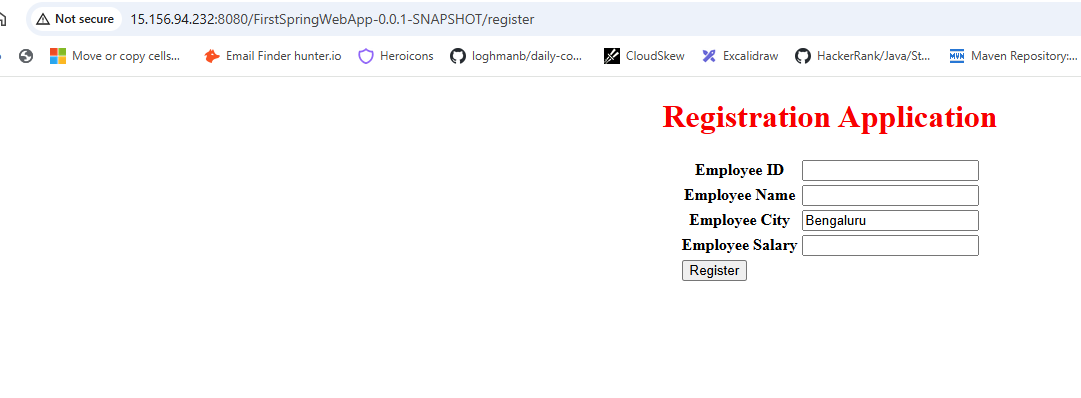




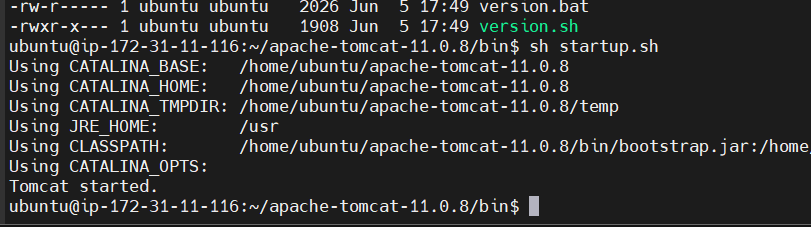
<http://15.156.94.232:8080/manager/html>

Go to Tomcat ---> Manage Apps

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

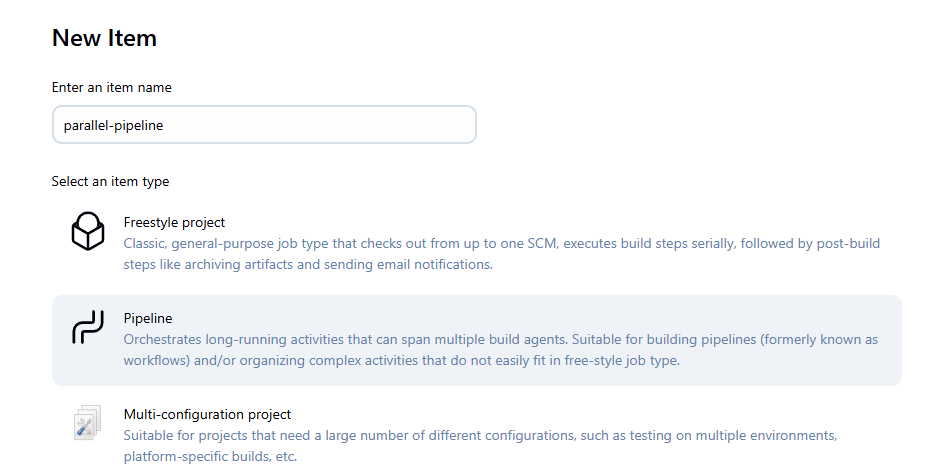


1:30



Parallel Stages:

Create a new pipeline



Just get any of the script to add into this pipeline

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'

}

}

}

}

Modified script

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('parallel stage') {

parallel {

stage('code-review'){

steps {

echo 'code review'

}

}

stage('nexus-upload'){

steps {

echo 'nexus upload'

}

}

}

}

stage('app deployed') {

steps {

echo 'Deploying App with Tomcat'

}

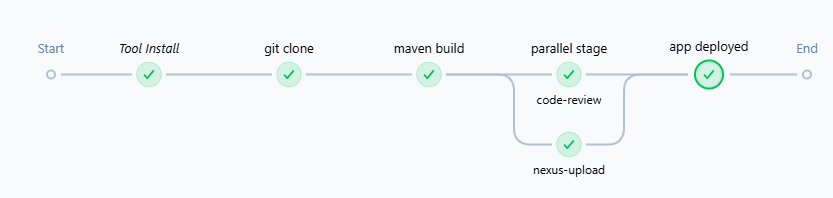
}

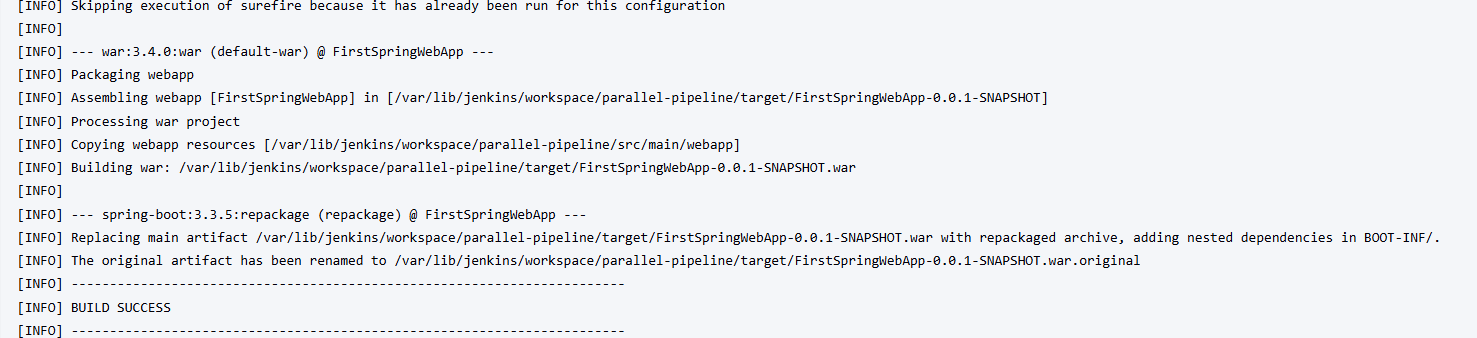
}

}

Apply and Save

Build Now





Code review and nexus-upload execute parallely

Shared library

Artifact upload

Core-review

mavenbuild

Pipeline 1

Pipeline 2

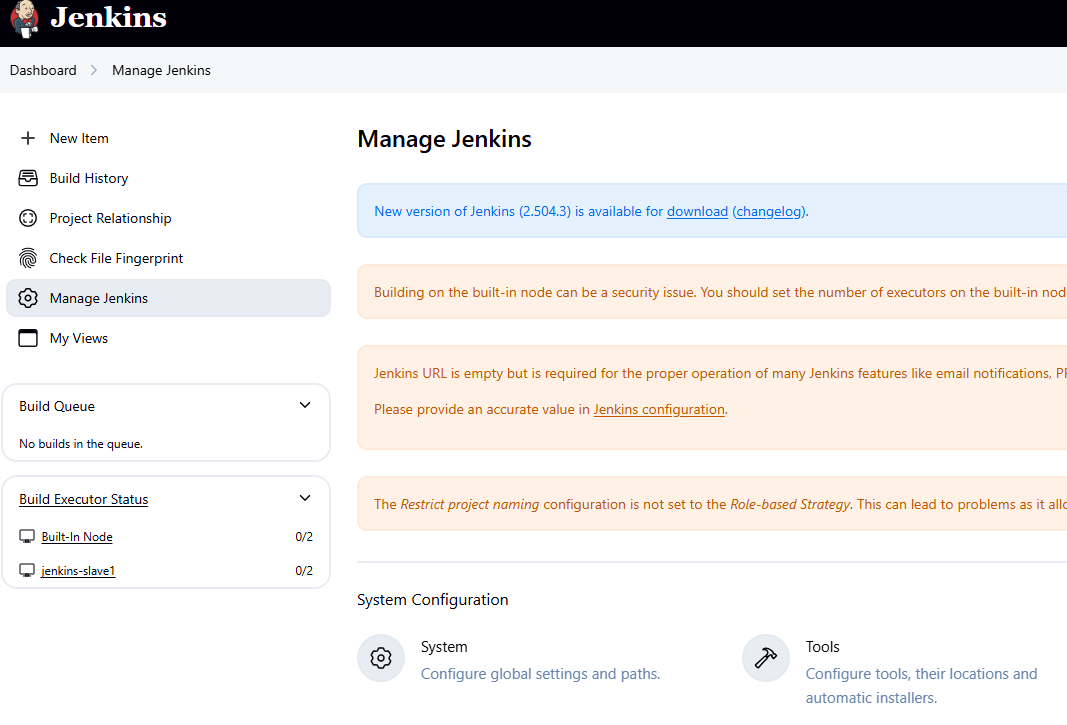
Shared library

Pipeline 3

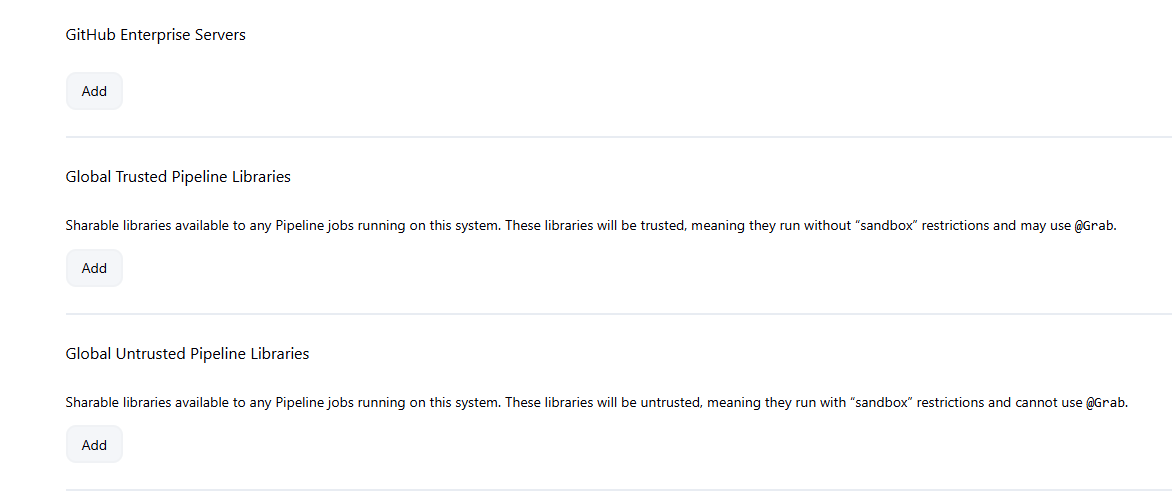
How do we define Groovy syntax in Jenkins?

Lets say code-review code is the same, nexus-upload code is the same, why to write the same code again and again

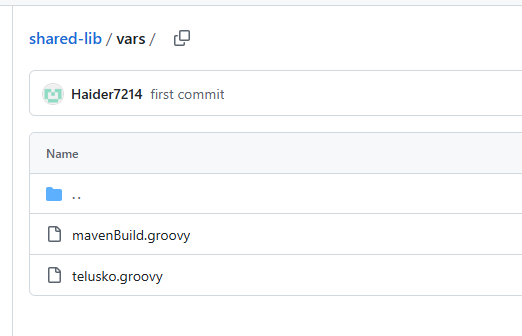
Manage Jenkins --> System

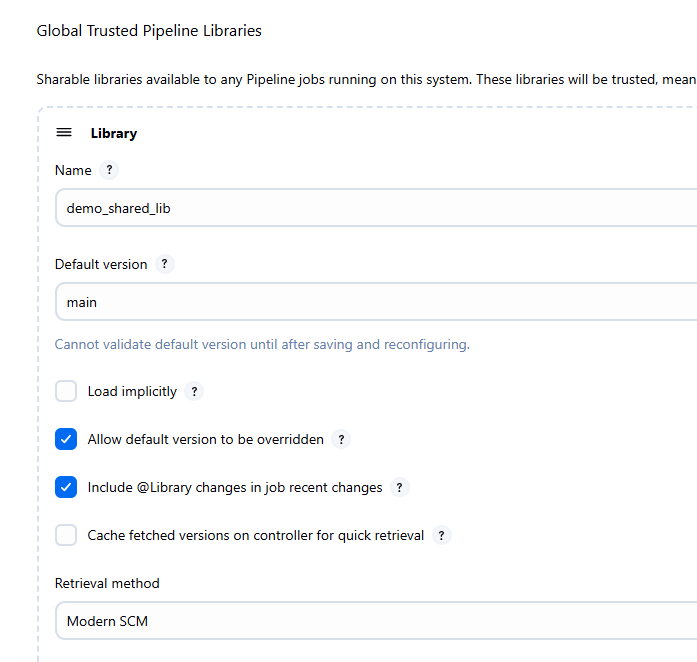


Global Trusted Pipeline Libraries

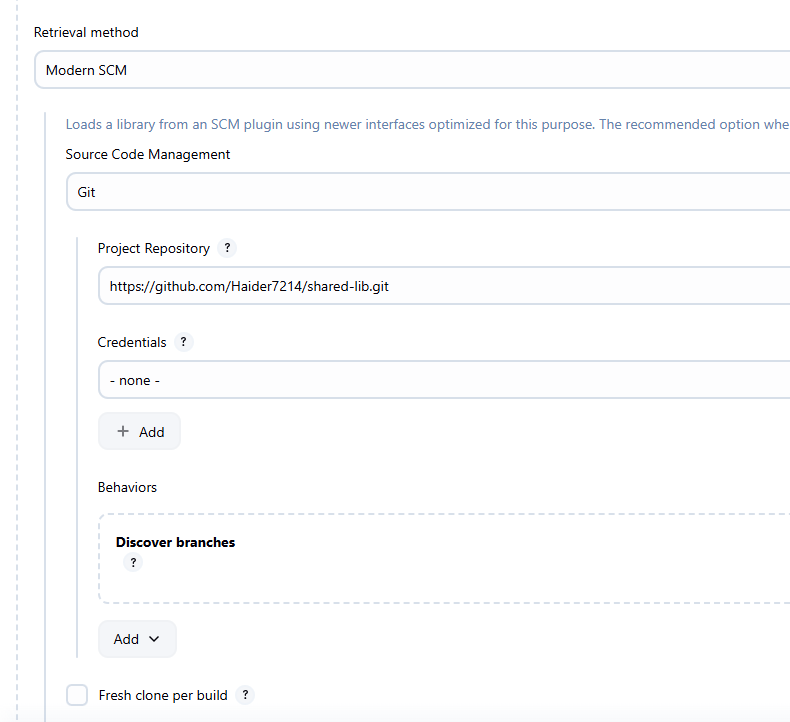


Github groovy scripts are there

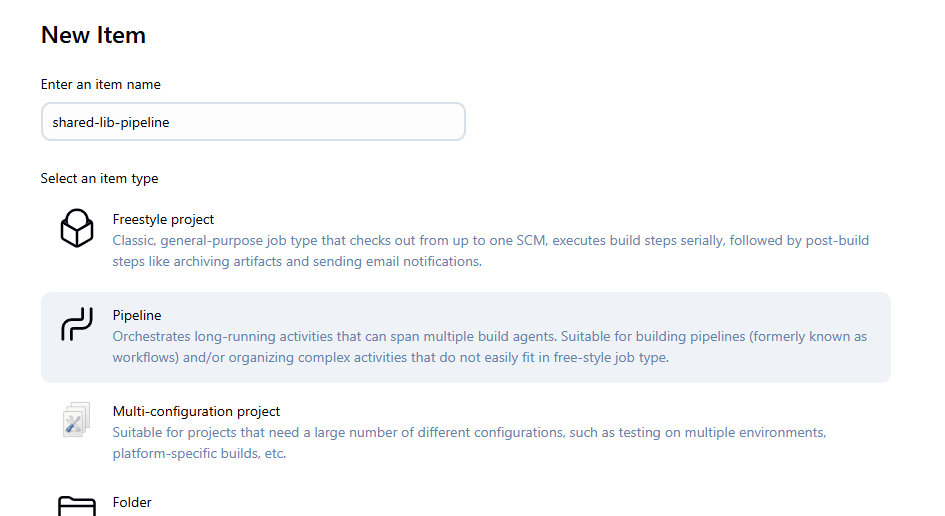




Credentials are not needed because it is a Public repo



Apply and Save



@Library('demo\_shared\_lib')

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('demo message') {

steps {

telusko()

}

}

stage('maven build') {

steps {

mavenBuild()

}

}

stage('parallel stage') {

parallel {

stage('code-review'){

steps {

echo 'code review'

}

}

stage('nexus-upload'){

steps {

echo 'nexus upload'

}

}

}

}

stage('app deployed') {

steps {

echo 'Deploying App with Tomcat'

}

}

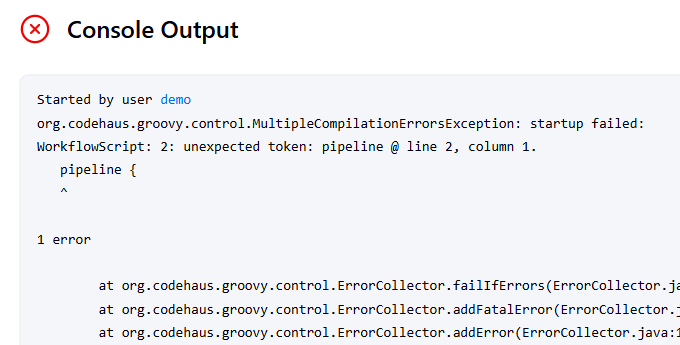
}

}

Apply and Save

Build Now

It failed



Added an \_

@Library('demo\_shared\_lib')\_

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('demo message') {

steps {

telusko()

}

}

stage('maven build') {

steps {

mavenBuild()

}

}

stage('parallel stage') {

parallel {

stage('code-review'){

steps {

echo 'code review'

}

}

stage('nexus-upload'){

steps {

echo 'nexus upload'

}

}

}

}

stage('app deployed') {

steps {

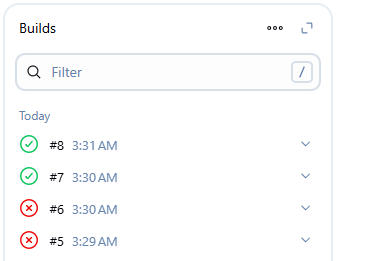
echo 'Deploying App with Tomcat'

}

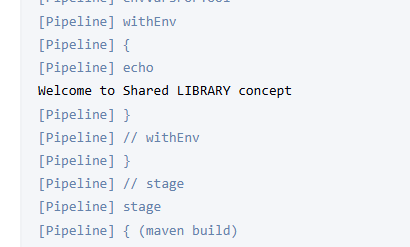
}

}

}



Function runs fine



Pipeline overview

