JENKINS PIPELINE JOBS

1) Create one Declarative pipeline job https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git

2) Create one scripted pipeline job https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git

3) Create one parameterized job https://github.com/betawins/spring3-mvc-maven-xml-hello-world-1.git

4) Write sample skeleton for declarative and scripted pipelines.

5) Create one declarative job https://github.com/betawins/sabear_simplecutomerapp/tree/feature-1.1

6) Create one declarative job https://github.com/betawins/Trading-UI.git

7) Execute parallel stages using Jenkins pipeline for any sample job

8) Execute Jenkins pipeline stages using shared libraries. Ref: https://phoenixnap.com/kb/jenkins-shared-library

9) Create One Jenkins job to Build and push the Docker image to Docker Hub. (https://github.com/betawins/Python-app.git)

CREATING JENKINS SERVER

→ LAUNCH ONE EC2 WITH USERDATA WHICH WILL INSTALL JENKINS IN THAT SERVER WITH ALLOWED 8080 PORT

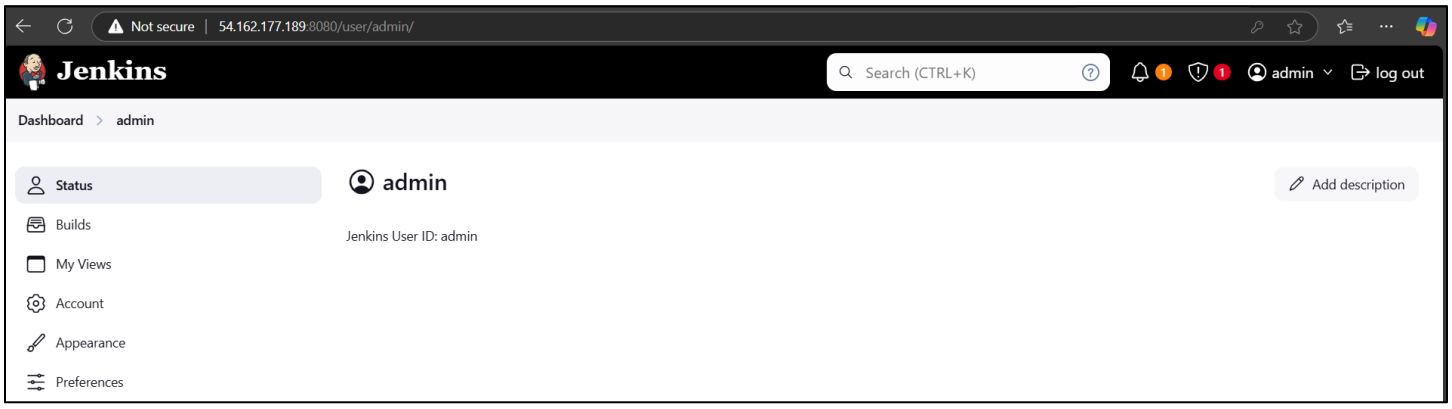
```
#!/bin/bash
sudo yum update -y
sudo amazon-linux-extras install java-17* -y
sudo yum install -y git

# Add Jenkins repository and install Jenkins
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade -y

# Add required dependencies for the jenkins package
sudo yum install fontconfig java-17-openjdk
sudo yum install jenkins -y
sudo systemctl daemon-reload

# Start and enable Jenkins service
sudo systemctl start jenkins
sudo systemctl enable jenkins

# Open port 8080 for Jenkins
sudo firewall-cmd --permanent --zone=public --add-port=8080/tcp
sudo firewall-cmd --reload
```

→ CONNECT TO SERVER AND CHECK VERSION OF JAVA , GIT AND JENKINS ALSO MAKE SURE IT IS UP AND RUNNING

```
[ec2-user@ip-10-0-0-17 ~]$ java --version
openjdk 17.0.13 2024-10-15 LTS
OpenJDK Runtime Environment Corretto-17.0.13.11.1 (build 17.0.13+11-LTS)
OpenJDK 64-Bit Server VM Corretto-17.0.13.11.1 (build 17.0.13+11-LTS, mixed mode, sharing)
[ec2-user@ip-10-0-0-17 ~]$ git --version
git version 2.40.1
[ec2-user@ip-10-0-0-17 ~]$ jenkins --version
2.479.2
[ec2-user@ip-10-0-0-17 ~]$ systemctl status jenkins | grep Active
   Active: active (running) since Wed 2024-12-25 13:06:55 UTC; 1min 48s ago
```

→ LOGIN TO JENKINS THROUGH InitialAdminPassword AND MAKE IT SECURE BY ADDING PASSWORD

sudo cat /var/lib/jenkins/secrets/initialAdminPassword



1) Create one Declarative pipeline job

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

fork the repo and make required changes in Jenkinsfile
Jenkins file in master branch in repo with declarative **PIPELINE CONSISTS OF THREE STAGES**

→**GIT CLONE**
→**MAVEN BUILD**
→**NEXUS**
→LAUNCH ONE EC2 WITH USERDATA WHICH WILL INSTALL NEXUS IN THAT SERVER

```
#!/bin/bash

sudo yum install -y java-1.8*
cd /opt
sudo wget https://sonatype-download.global.ssl.fastly.net/nexus/3/nexus-3.0.2-02-unix.tar.gz
sudo tar -zxvf nexus-3.0.2-02-unix.tar.gz
sudo mv /opt/nexus-3.0.2-02 /opt/nexus
sudo adduser nexus
echo "nexus ALL=(ALL) NOPASSWD: ALL" | sudo tee -a /etc/sudoers
sudo chown -R nexus:nexus /opt/nexus
echo "run_as_user=\"nexus\"" | sudo tee /opt/nexus/bin/nexus.rc
sudo ln -s /opt/nexus/bin/nexus /etc/init.d/nexus
sudo -u nexus /opt/nexus/bin/nexus start
sudo chkconfig nexus on
```

→CONNECT TO PORT 8081 FOLLOWED BY SERVER PUBLIC IP AND CHECK VERSION OF NEXUS
USERNAME = admin PASSWORD = admin123

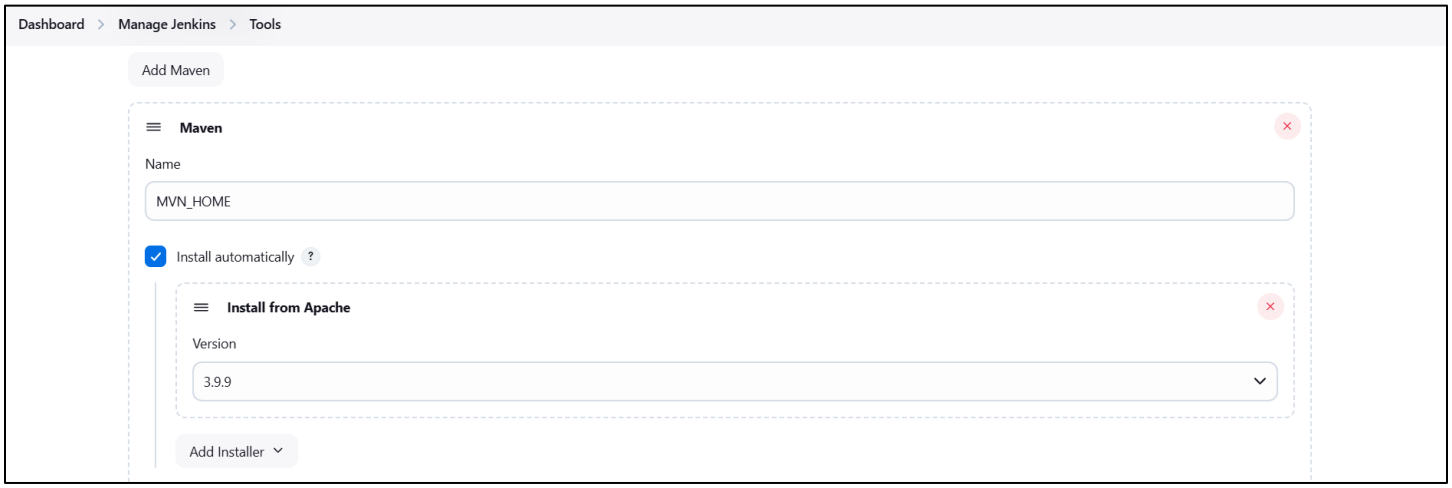
Create a repo for package

settings > repositories > create repository > name > configurations > create

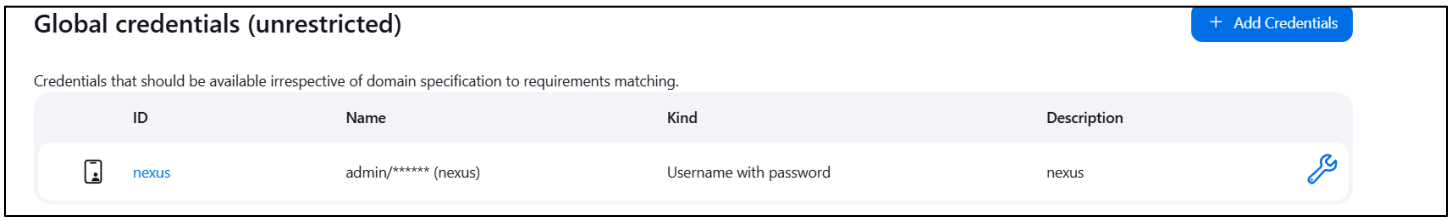
<make sure to install plugins MAVEN, NEXUS, PIPELINE UTILITY STEPS>

1. Ensure Jenkins Configuration:

- **Maven Tool**: Ensure that Maven is configured in Jenkins. Go to **Manage Jenkins** > **Global Tool Configuration** and configure Maven with the name MVN_HOME.



- **Nexus Credentials**: Add the Nexus server credentials in Jenkins. Go to **Manage Jenkins** > **Manage Credentials** and add a new credential with the ID Nexus_server.



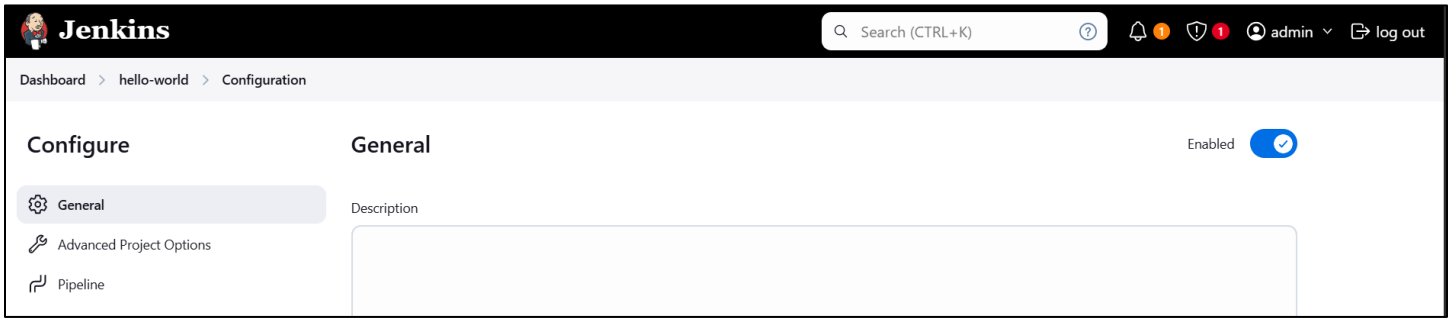
UPDATED SCRIPT FOR DECLARATIVE PIPELINE

```
pipeline {
  agent any
  tools {
    maven "MVN_HOME"
  }
  environment {
    NEXUS_VERSION = "nexus3"
    NEXUS_PROTOCOL = "http"
    NEXUS_URL = "54.90.131.74:8081"
    NEXUS_REPOSITORY = "spring3"
    NEXUS_CREDENTIAL_ID = "nexus"
    PACKAGING = "war"  // Manually set the packaging type
  }
  stages {
    stage("Clone Code") {
      steps {
        script {
          git 'https://github.com/Farsaan-tech/spring3-mvc-maven-xml-hello-world-1.git'
        }
      }
    }
    stage("Build with Maven") {
      steps {
        script {
```

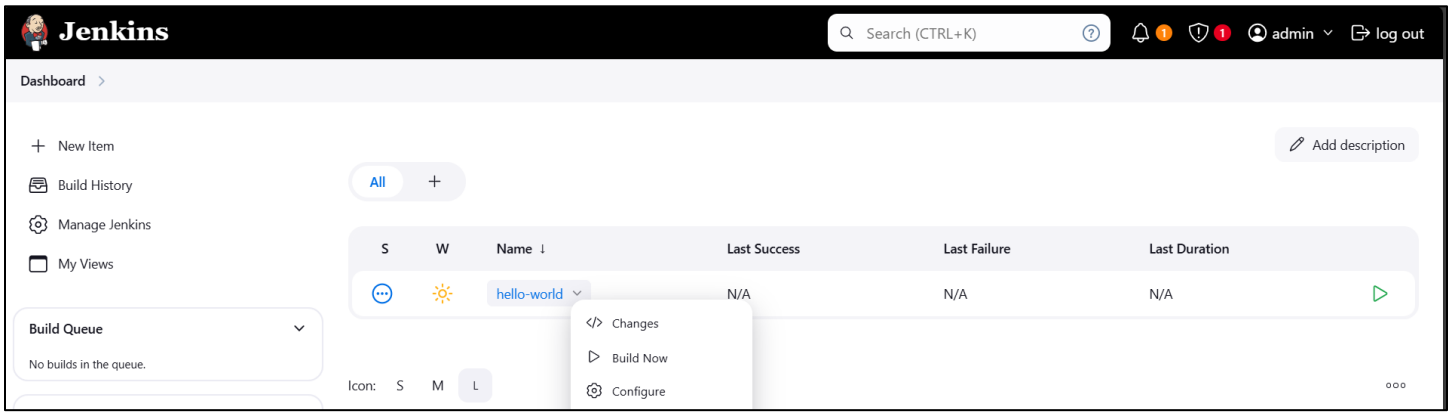
```
                sh 'mvn -Dmaven.test.failure.ignore=true install'
            }
        }
    }
    stage("Publish to Nexus") {
        steps {
            script {
                // Manually set the path of the artifact based on the packaging type
                def artifactPath = "target/${env.PACKAGING}-artifact.${env.PACKAGING}"
                // Check if the artifact exists
                artifactExists = fileExists artifactPath
                if (artifactExists) {
                    echo "*** File: ${artifactPath}, version ${BUILD_NUMBER}"
                    nexusArtifactUploader(
                        nexusVersion: env.NEXUS_VERSION,
                        protocol: env.NEXUS_PROTOCOL,
                        nexusUrl: env.NEXUS_URL,
                        groupId: 'com.ncodeit',
                        version: "${BUILD_NUMBER}",
                        repository: env.NEXUS_REPOSITORY,
                        credentialsId: env.NEXUS_CREDENTIAL_ID,
                        artifacts: [
                            [artifactId: 'ncodeit-hello-world', classifier: '', file: artifactPath, type: env.PACKAGING],
                            [artifactId: 'ncodeit-hello-world', classifier: '', file: "pom.xml", type: "pom"]
                        ]
                    )
                } else {
                    error "*** File: ${artifactPath}, could not be found"
                }
            }
        }
    }
  }
}
```

Create a jenkins job

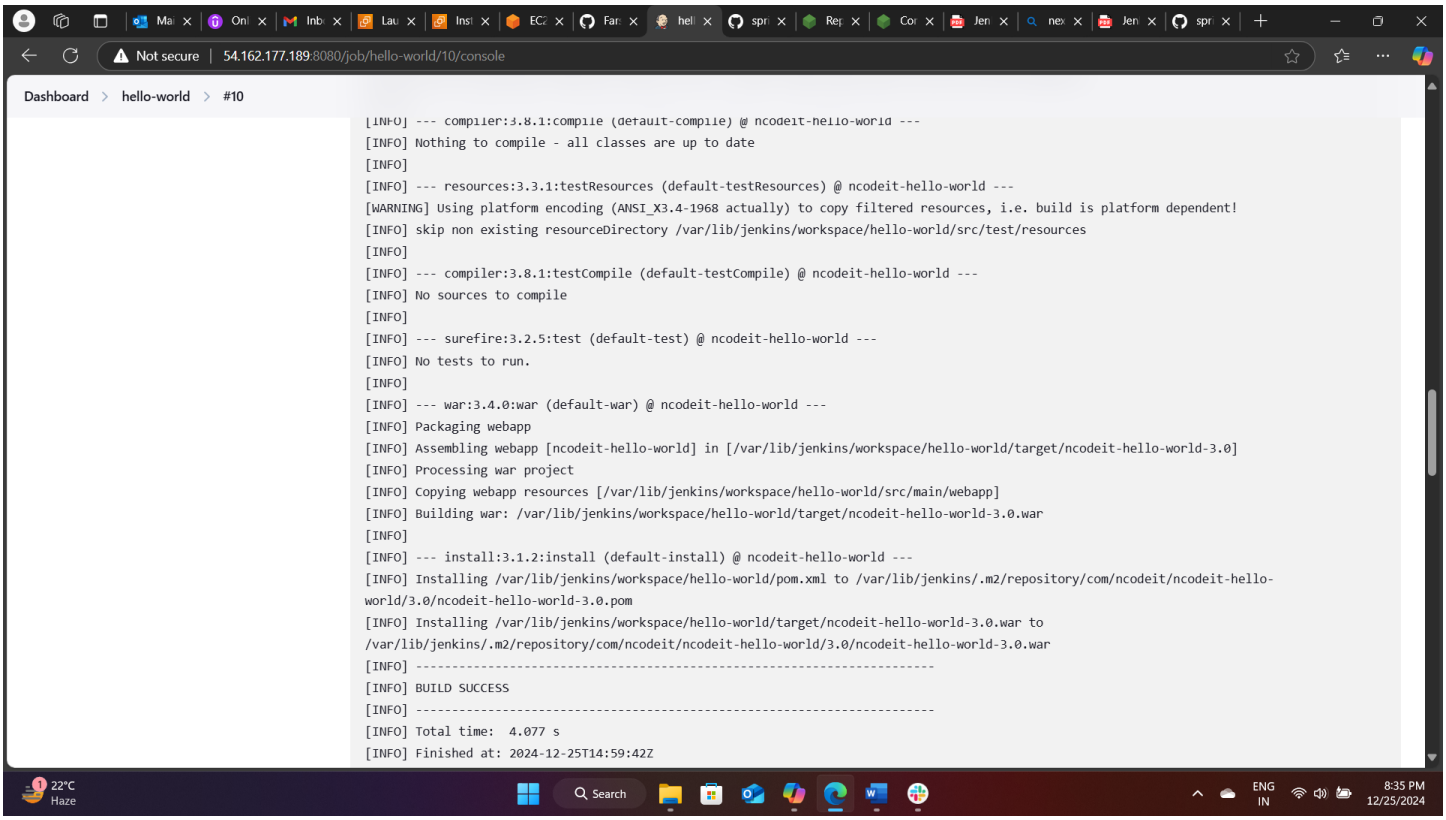
Dashboard > new item > item name > select pipeline > ok



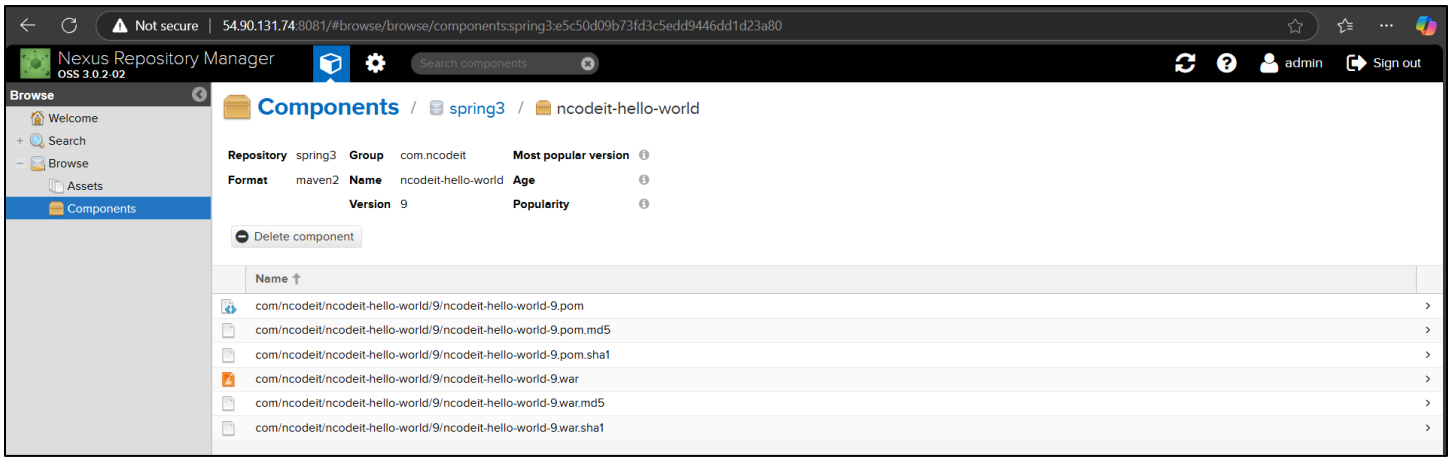
Pipeline > pipeline script from SCM > SCM > git > Repository URL > https://github.com/Farsaan-tech/spring3-mvc-maven-xml-hello-world-1.git > Branch > Master > script path > Jenkinsfile > Save



→Build Now



→ Verify on Nexus



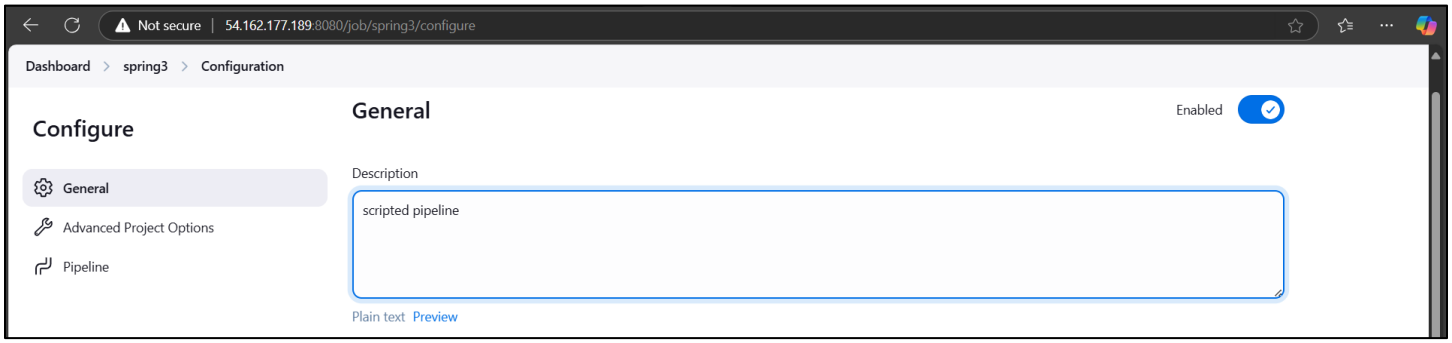
2) CREATE ONE SCRIPTED PIPELINE JOB

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

→create new repository in nexus and replace the name in environment in script

Create a jenkins job

Dashboard > new item > item name > select pipeline > ok

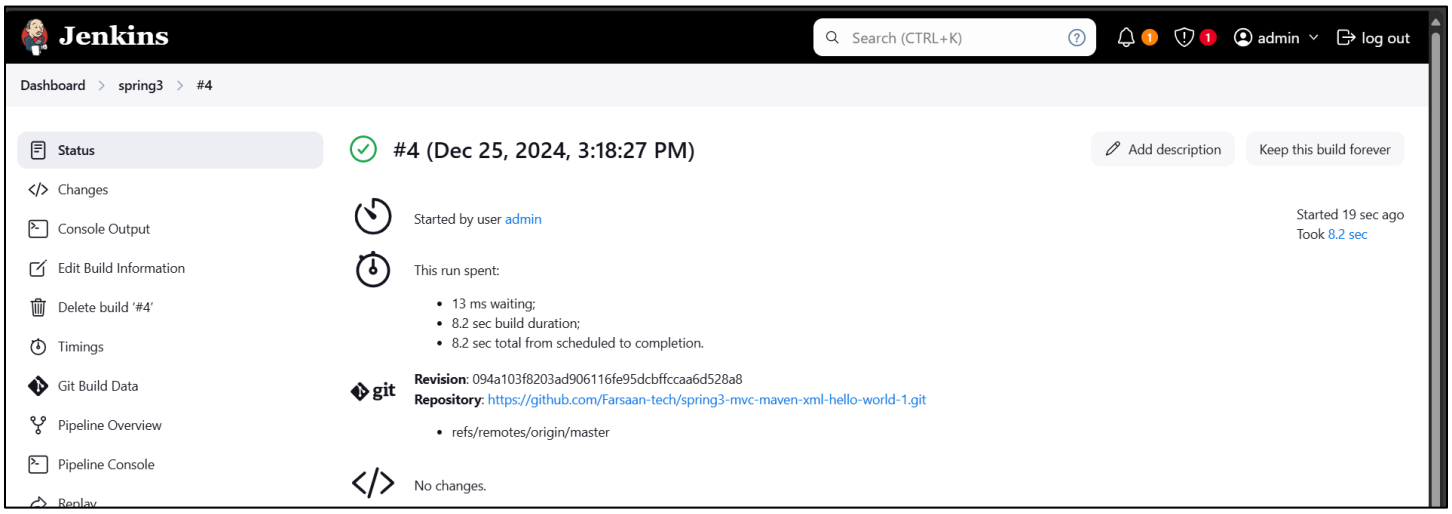


Pipeline > pipeline script > add the below script > save

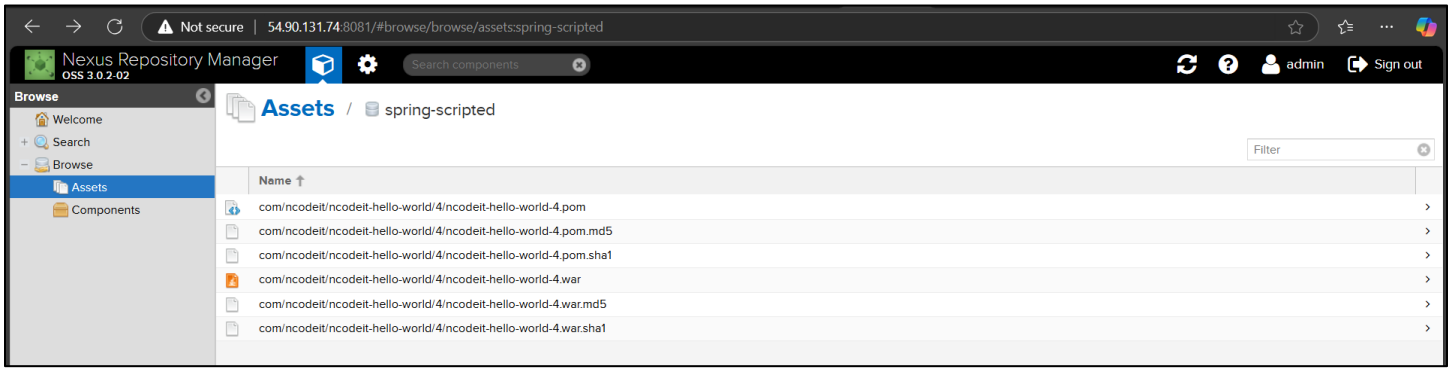
```
node {
  // Define tools
  def mvnHome = tool name: 'MVN_HOME', type: 'maven'

  // Set environment variables
  env.NEXUS_VERSION = "nexus3"
  env.NEXUS_PROTOCOL = "http"
  env.NEXUS_URL = "54.90.131.74:8081"
  env.NEXUS_REPOSITORY = "spring-scripted"
  env.NEXUS_CREDENTIAL_ID = "nexus"
  env.PACKAGING = "war"  // Manually set the packaging type

  stage('Clone Code') {
    checkout scm: [
      $class: 'GitSCM',
      branches: [[name: '*/master']],
      userRemoteConfigs: [[url: 'https://github.com/Farsaan-tech/spring3-mvc-maven-xml-hello-world-1.git']]
    ]
  }

  stage('Build with Maven') {
    sh "${mvnHome}/bin/mvn -Dmaven.test.failure.ignore=true install"
  }

  stage('Verify Build Output') {
    script {
      // List the contents of the target directory
      sh 'ls -al target/'
    }
  }

  stage('Publish to Nexus') {
    script {
      // Verify the actual artifact name and location
```

```
        def artifactPath = "target/ncodeit-hello-world-3.0.${env.PACKAGING}"
        artifactExists = fileExists artifactPath
        if (artifactExists) {
          echo "*** File: ${artifactPath}, version ${BUILD_NUMBER}"
          nexusArtifactUploader(
            nexusVersion: env.NEXUS_VERSION,
            protocol: env.NEXUS_PROTOCOL,
            nexusUrl: env.NEXUS_URL,
            groupId: 'com.ncodeit',
            version: "${BUILD_NUMBER}",
            repository: env.NEXUS_REPOSITORY,
            credentialsId: env.NEXUS_CREDENTIAL_ID,
            artifacts: [
              [artifactId: 'ncodeit-hello-world', classifier: '', file: artifactPath, type: env.PACKAGING],
              [artifactId: 'ncodeit-hello-world', classifier: '', file: "pom.xml", type: "pom"]
            ]
          )
        } else {
          error "*** File: ${artifactPath}, could not be found"
        }
      }
    }
}
```

→Build Now



→ Verify on Nexus



3) CREATE ONE PARAMETERIZED JOB

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

dashboard > new item > name > pipeline

In the **General** section, check the box **This project is parameterized**
Add String Parameters:

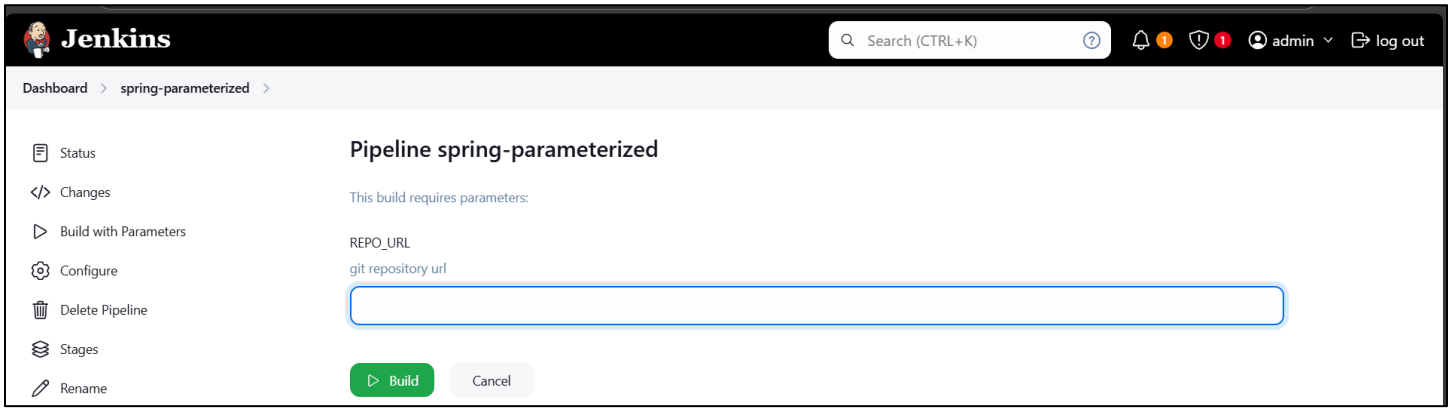
- Click **Add Parameter** and select **String Parameter**.
- Add the following parameters:
  - **Name**: `REPO_URL`
  - **Description**: `The GitHub repository URL to clone`

Add the below script to pipeline

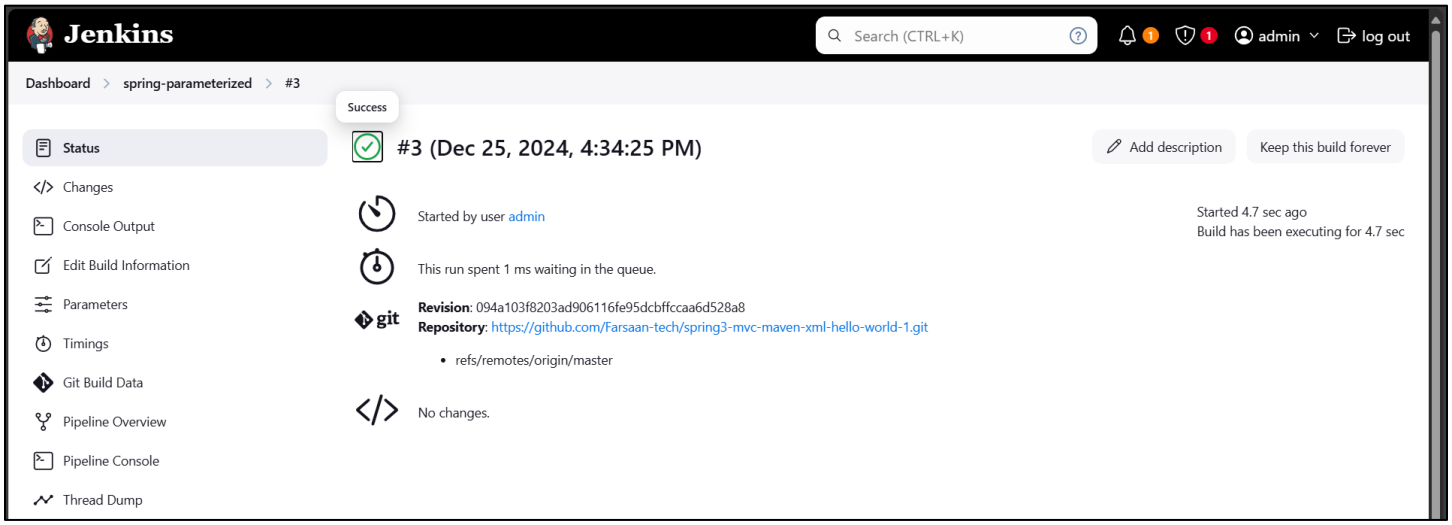
```
pipeline {
    agent any
    parameters {
        string(name: 'REPO_URL', description: 'The GitHub repository URL to clone.')
    }
    tools {
        maven "MVN_HOME"
    }
    environment {
        NEXUS_VERSION = "nexus3"
        NEXUS_PROTOCOL = "http"
        NEXUS_URL = "54.90.131.74:8081"
        NEXUS_REPOSITORY = "spring-scripted"
        NEXUS_CREDENTIAL_ID = "nexus"
        PACKAGING = "war"  // Manually set the packaging type
    }
    stages {
        stage('Clone Code') {
            steps {
                git url: "${params.REPO_URL}", branch: 'master'
            }
        }
        stage('Build with Maven') {
            steps {
                sh 'mvn -Dmaven.test.failure.ignore=true install'
            }
        }
        stage('Verify Build Output') {
            steps {
                sh 'ls -al target/'
            }
        }
        stage('Publish to Nexus') {
            steps {
                script {
                    def artifactPath = "target/ncodeit-hello-world-3.0.${env.PACKAGING}"
                    if (fileExists(artifactPath)) {
                        echo "*** File: ${artifactPath}, version ${BUILD_NUMBER}"
                        nexusArtifactUploader(
                            nexusVersion: env.NEXUS_VERSION,
                            protocol: env.NEXUS_PROTOCOL,
                            nexusUrl: env.NEXUS_URL,
```

```
                    groupId: 'com.ncodeit',
                    version: "${BUILD_NUMBER}",
                    repository: env.NEXUS_REPOSITORY,
                    credentialsId: env.NEXUS_CREDENTIAL_ID,
                    artifacts: [
                        [artifactId: 'ncodeit-hello-world', classifier: '', file: artifactPath, type: env.PACKAGING],
                        [artifactId: 'ncodeit-hello-world', classifier: '', file: "pom.xml", type: "pom"]
                    ]
                )
            } else {
                error "*** File: ${artifactPath}, could not be found"
            }
        }
    }
}
}
}
```

→BUILD WITH PARAMETERS > ADD THE URL OF GITHUB REPO



→BUILD



4) Write sample skeleton for declarative and scripted pipelines.

```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        echo 'Building...'
      }
    }
    stage('Test') {
      steps {
        echo 'Testing...'
      }
    }
    stage('Deploy') {
      steps {
        echo 'Deploying...'
      }
    }
  }
```

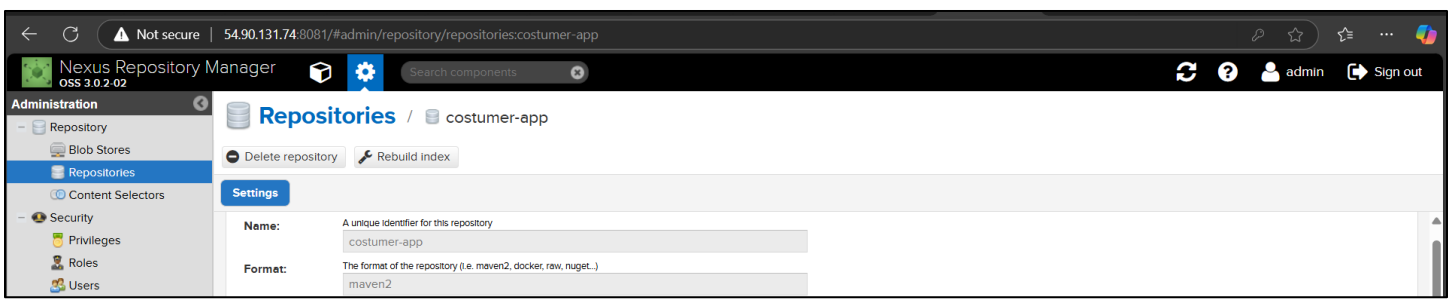
```
node {
  stage('Build') {
    echo 'Building...'
  }
  stage('Test') {
    echo 'Testing...'
  }
  stage('Deploy') {
    echo 'Deploying...'
  }
  // Post actions
  always {
    echo 'This will always run'
  }
  success {
    echo 'This will run only if successful'
  }
  failure {
    echo 'This will run only if failed'
  }
}
```

5) Create one declarative job

https://github.com/betawins/sabear-simplecutomerapp/tree/feature-1.1

fork the repo and make required changes in Jenkinsfile and create a repo in nexus

(SUCH AS NEXUS CREDENTIALS . NEXUS URL , NEXUS REPO NAME , GIT URL)



Jenkins file in master branch in repo with declarative **PIPELINE CONSISTS OF THREE STAGES**

→**GIT CLONE**

→**MAVEN BUILD**

→**NEXUS**

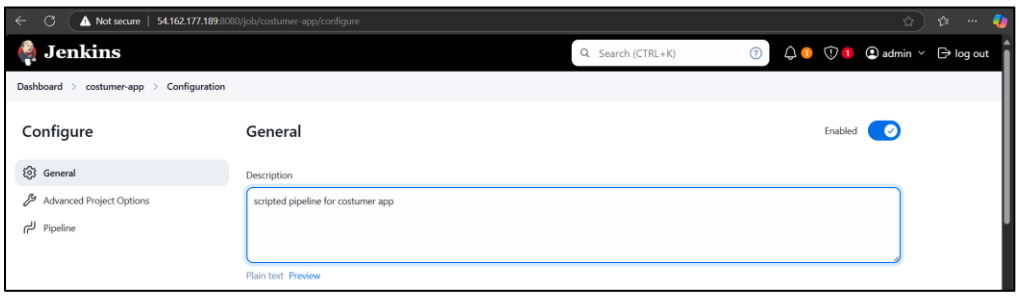
UPDATED SCRIPT FOR DECLARATIVE PIPELINE

```
pipeline {
  agent {
    label "master"
  }
  tools {
    maven "MVN_HOME"
  }
  environment {
    NEXUS_VERSION = "nexus3"
```

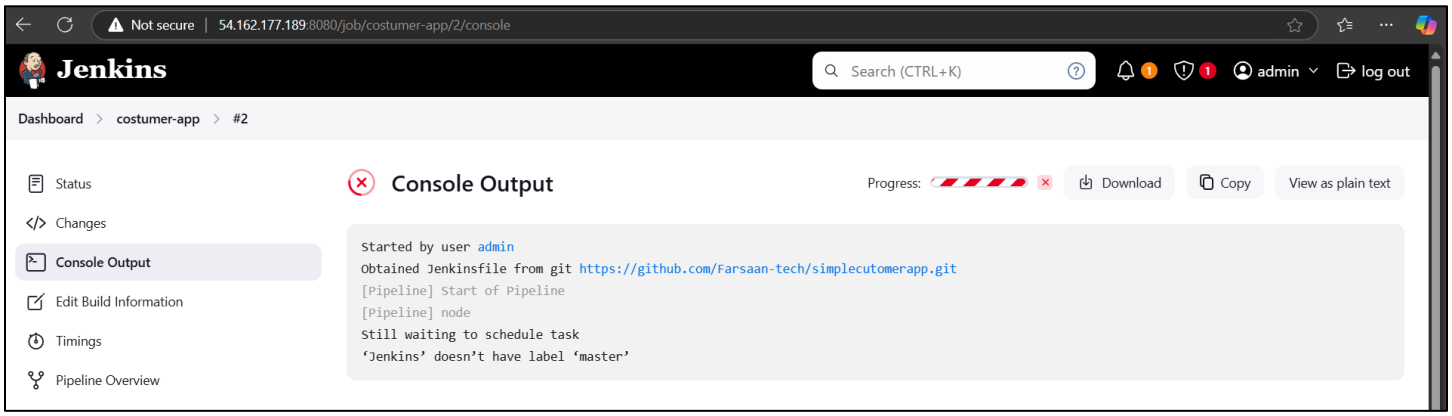
```
        NEXUS_PROTOCOL = "http"
        NEXUS_URL = "54.90.131.74:8081"
        NEXUS_REPOSITORY = "costumer-app"
        NEXUS_CREDENTIAL_ID = "nexus"
    }
    stages {
        stage("clone code") {
            steps {
                script {
                    git ' https://github.com/Farsaan-tech/simplecutomerapp.git '
                }
            }
        }
        stage("mvn build") {
            steps {
                script {
                    sh 'mvn -Dmaven.test.failure.ignore=true install'
                }
            }
        }
        stage("publish to nexus") {
            steps {
                script {
                    pom = readMavenPom file: "pom.xml"
                    filesByGlob = findFiles(glob: "target/*.${pom.packaging}")
                    echo "${filesByGlob[0].name} ${filesByGlob[0].path} ${filesByGlob[0].directory} ${filesByGlob[0].length}
${filesByGlob[0].lastModified}"
                    artifactPath = filesByGlob[0].path
                    artifactExists = fileExists artifactPath
                    if (artifactExists) {
                        echo "*** File: ${artifactPath}, group: ${pom.groupId}, packaging: ${pom.packaging}, version ${pom.version}"
                        nexusArtifactUploader(
                            nexusVersion: NEXUS_VERSION,
                            protocol: NEXUS_PROTOCOL,
                            nexusUrl: NEXUS_URL,
                            groupId: pom.groupId,
                            version: pom.version,
                            repository: NEXUS_REPOSITORY,
                            credentialsId: NEXUS_CREDENTIAL_ID,
                            artifacts: [
                                [artifactId: pom.artifactId, classifier: '', file: artifactPath, type: pom.packaging],
                                [artifactId: pom.artifactId, classifier: '', file: "pom.xml", type: "pom"]
                            ]
                        )
                    } else {
                        error "*** File: ${artifactPath}, could not be found"
                    }
                }
            }
        }
    }
}
```

Create a jenkins job

Dashboard > new item > item name > select pipeline > ok

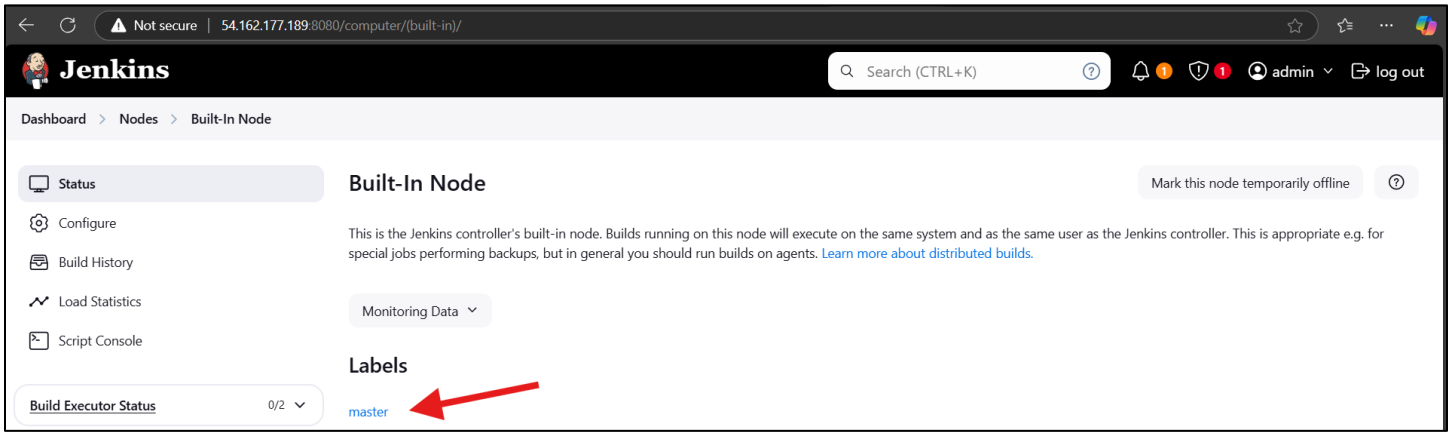


Pipeline > pipeline script from SCM > SCM > git > Repository URL > https://github.com/Farsaan-tech/simplecutomerapp.git > Branch > master > script path > Jenkinsfile > Save > Build Now

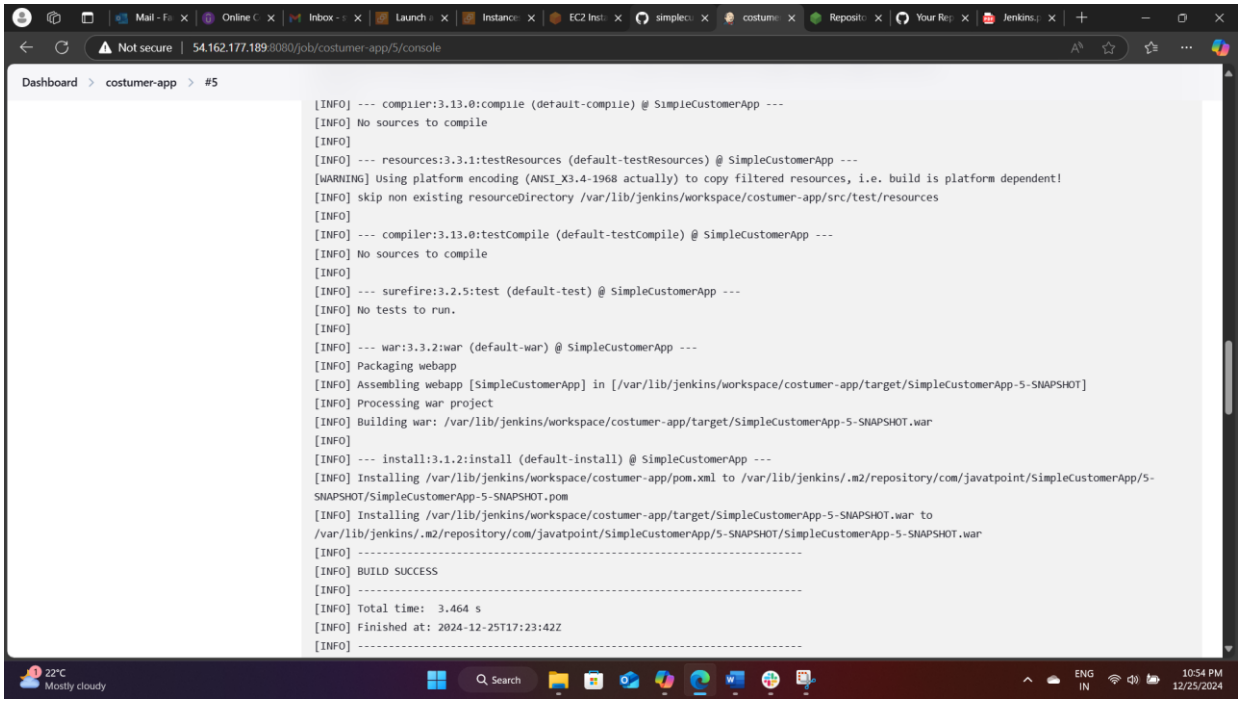


→ADD A NODE WITH LABEL MASTER

Dashboard > manage Jenkins > Nodes > built in node > configure > label > master > save



→ check the console output



note= CHECK THE CONSOLE OUTPUT IT MAY REQUIRE SOME ADMIN APPROVALS

6) CREATE ONE DECLARATIVE JOB

https://github.com/betawins/Trading-UI.git

fork the repo

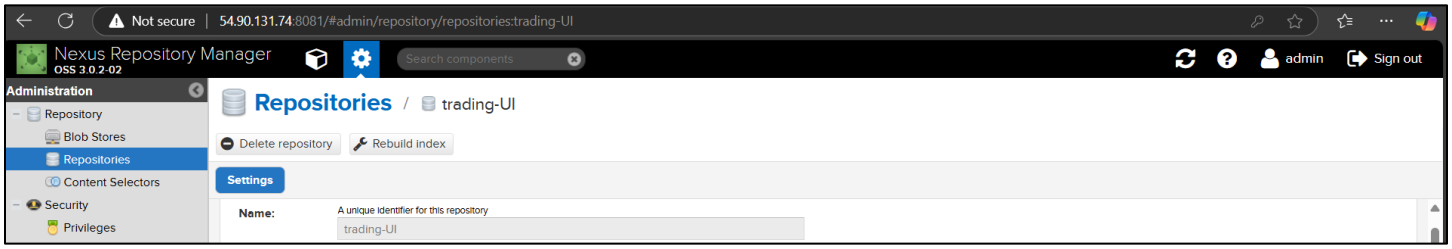
→ This repository is missing pom.xml file

Create a pom.xml file in forked repository

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>trading-ui</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <name>Trading UI</name>
    <url>http://maven.apache.org</url>

    <properties>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- Add your project dependencies here -->
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>${maven.compiler.source}</source>
                    <target>${maven.compiler.target}</target>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.3</version>
                <configuration>
                    <warName>trading-ui</warName>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

make required changes in Jenkinsfile and create a repo in nexus

(SUCH AS NEXUS CREDENTIALS . NEXUS URL , NEXUS REPO NAME , GIT URL)



Jenkins file in master branch in repo with declarative **PIPELINE CONSISTS OF THREE STAGES**

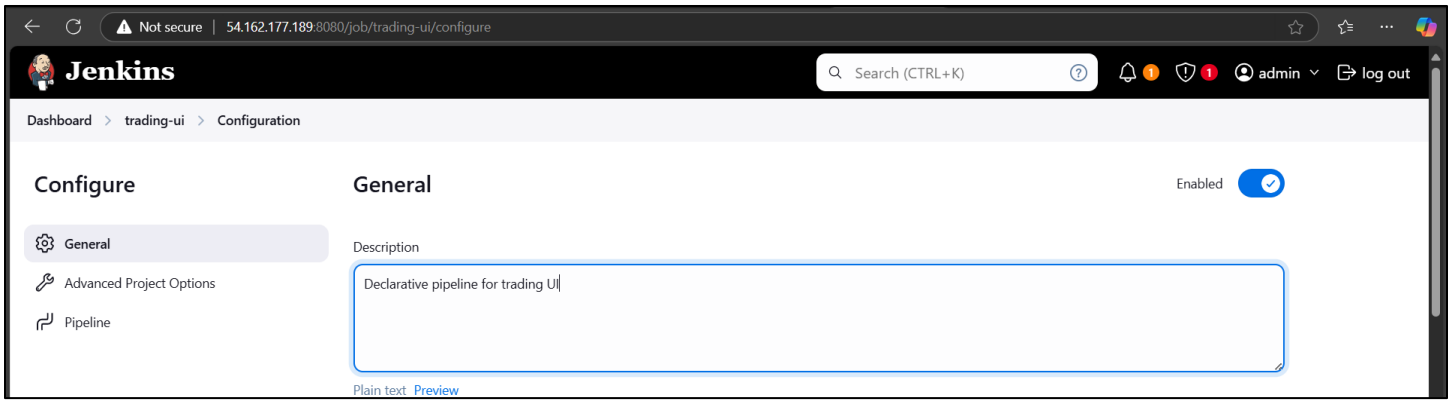
→**GIT CLONE**

→**MAVEN BUILD**

→**NEXUS**

Create a jenkins job

Dashboard > new item > item name > select pipeline > ok



Pipeline > pipeline script > add the below script > save > Build now

UPDATED SCRIPT FOR DECLARATIVE PIPELINE

```
pipeline {
    agent any
    tools {
        maven "MVN_HOME"
    }
    environment {
        NEXUS_VERSION = "nexus3"
        NEXUS_PROTOCOL = "http"
        NEXUS_URL = "54.90.131.74:8081"
        NEXUS_REPOSITORY = "trading-UI"
        NEXUS_CREDENTIAL_ID = "nexus"
        PACKAGING = "war"
    }
    stages {
        stage('Clone Code') {
            steps {
                git url: 'https://github.com/Farsaan-tech/Trading-UI.git', branch: 'patch-1'
            }
        }
        stage('Build with Maven') {
            steps {
                script {
                    // Print the working directory and list files to verify pom.xml presence
                    sh 'pwd && ls -al'
                    // Build the project with Maven
```

```
                sh 'mvn clean install -Dmaven.test.failure.ignore=true'
            }
        }
    }
    stage('Verify Build Output') {
        steps {
            script {
                // List the target directory contents
                sh 'ls -al target/'
            }
        }
    }
    stage('Publish to Nexus') {
        steps {
            script {
                // Adjusted artifact path
                def artifactPath = "target/trading-ui.war"
                // Check if the artifact exists
                artifactExists = fileExists artifactPath
                if (artifactExists) {
                    echo "*** File: ${artifactPath}, version ${BUILD_NUMBER}"
                    nexusArtifactUploader(
                        nexusVersion: env.NEXUS_VERSION,
                        protocol: env.NEXUS_PROTOCOL,
                        nexusUrl: env.NEXUS_URL,
                        groupId: 'com.example',  // Replace with your actual groupId
                        version: "${BUILD_NUMBER}",
                        repository: env.NEXUS_REPOSITORY,
                        credentialsId: env.NEXUS_CREDENTIAL_ID,
                        artifacts: [
                            [artifactId: 'trading-ui', classifier: '', file: artifactPath, type: env.PACKAGING],
                            [artifactId: 'trading-ui', classifier: '', file: "pom.xml", type: "pom"]
                        ]
                    )
                } else {
                    error "*** File: ${artifactPath}, could not be found"
                }
            }
        }
    }
  }
}
```

→ CONSOLE OUTPUT

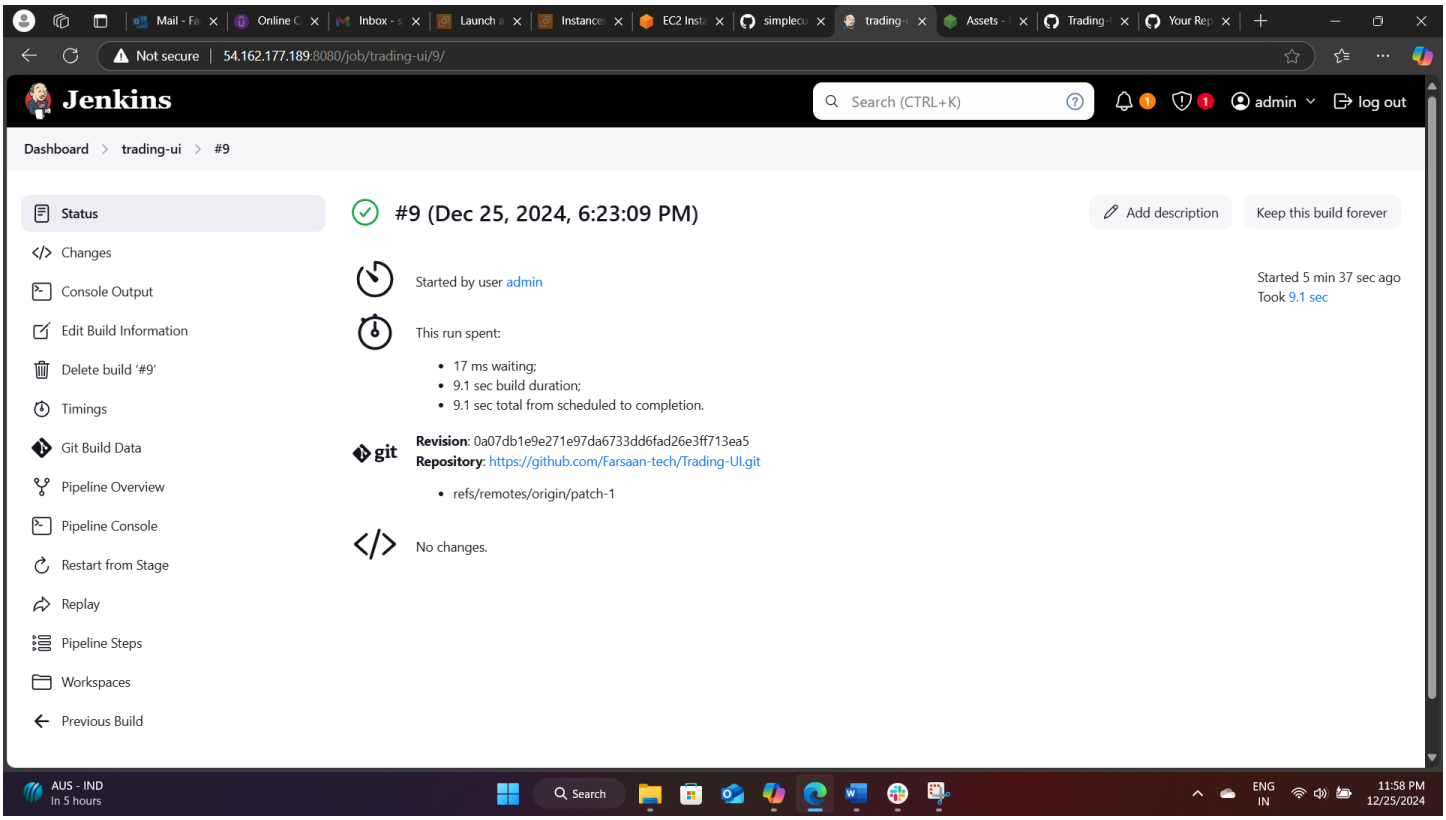
```
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-war-plugin:3.2.3:war (default-war) on project trading-ui: Error assembling WAR:
webxml attribute is required (or pre-existing WEB-INF/web.xml if executing in update mode) -> [Help 1]
```

→OUTPUT GIVES ERROR THAT web.xml IS NOT AVAILABLE LETS CREATE A BASIC FILE IN REPO WITH BELOW CONTENT

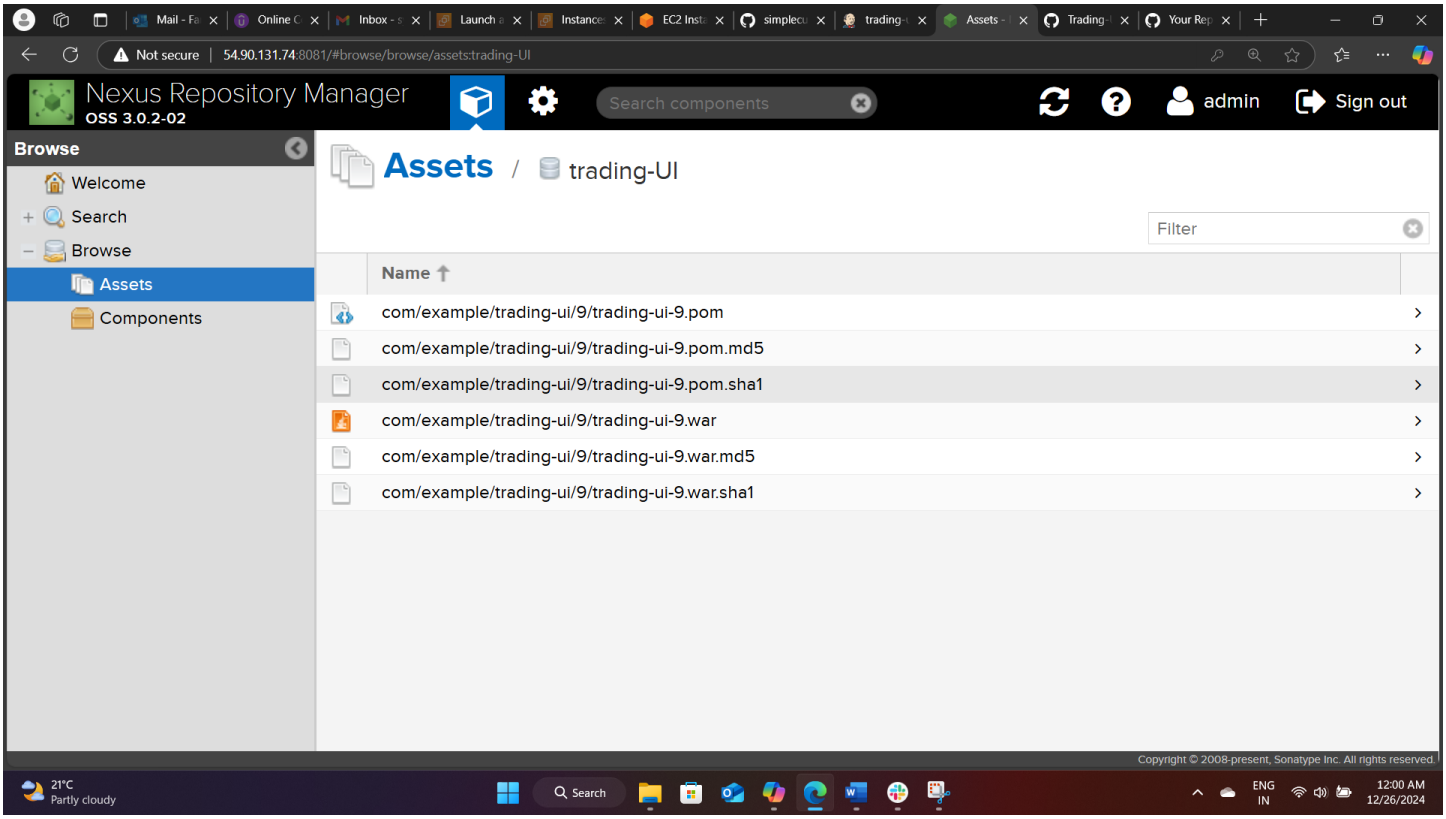
→ON GUI ADD FILE TO REPO FILE NAME **src/main/webapp/WEB-INF/web.xml**

```
<WEB-APP XMLNS="HTTP://XMLNS.JCP.ORG/XML/NS/JAVAEE"
    XMLNS:XSI="HTTP://WWW.W3.ORG/2001/XMLSCHEMA-INSTANCE"
    XSI:SCHEMALOCATION="HTTP://XMLNS.JCP.ORG/XML/NS/JAVAEE
                HTTP://XMLNS.JCP.ORG/XML/NS/JAVAEE/WEB-APP_3_1.XSD"
    VERSION="3.1">
  <DISPLAY-NAME>TRADING UI</DISPLAY-NAME>
</WEB-APP>
```

→BUILD NOW



→ NEXUS ARTIFACTORY



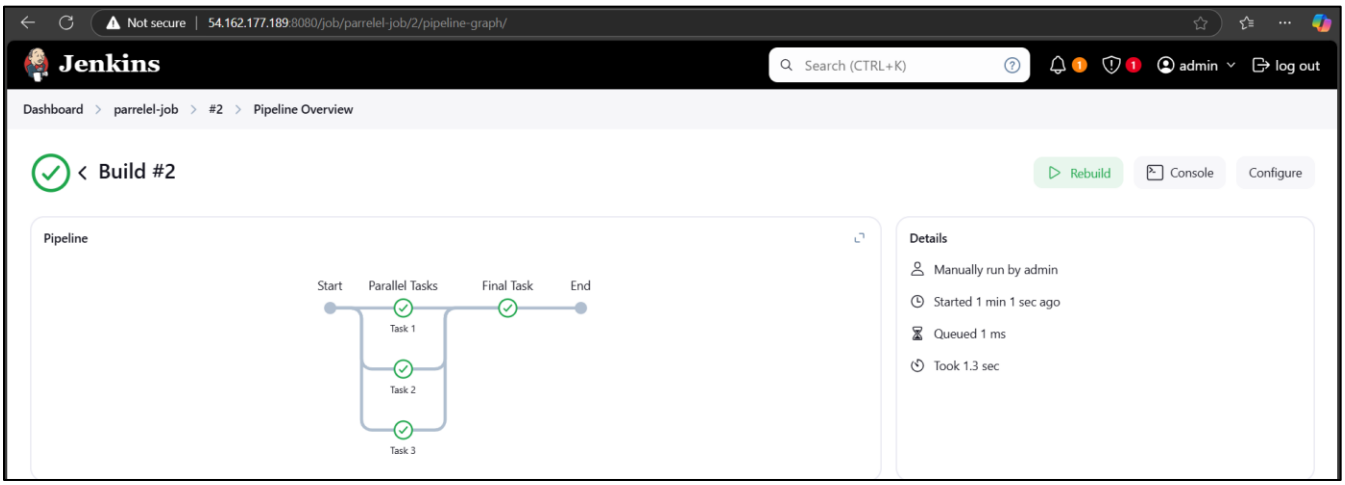
7) EXECUTE PARALLEL STAGES USING JENKINS PIPELINE FOR ANY SAMPLE JOB

Create a jenkins job

Dashboard > new item > item name > select pipeline > ok > add sample declarative pipeline > save > build

```
pipeline {
  agent any
  stages {
    stage('Parallel Tasks') {
      parallel {
        stage('Task 1') {
          steps {
            echo 'Running Task 1...'
            sh 'echo Task 1 is running'
          }
        }
        stage('Task 2') {
          steps {
            echo 'Running Task 2...'
            sh 'echo Task 2 is running'
          }
        }
        stage('Task 3') {
          steps {
            echo 'Running Task 3...'
            sh 'echo Task 3 is running'
          }
        }
      }
    }
    stage('Final Task') {
      steps {
        echo 'Running Final Task...'
        // Add additional commands here
        sh 'echo Final task is running'
      }
    }
  }
}
```

→ Dashboard > parrelel-job> Pipeline Overview



8) Execute Jenkins pipeline stages using shared libraries

What Is a Shared Library in Jenkins?

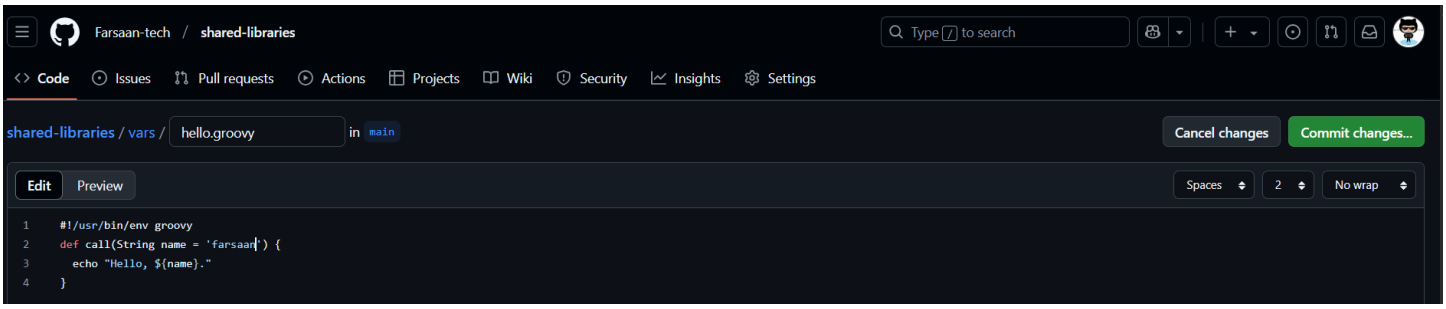
A shared library in Jenkins is a collection of Groovy scripts shared between different Jenkins jobs. To run the scripts, they are pulled into a Jenkinsfile.

Why Use a Jenkins Shared Library?

Developers use shared libraries to avoid writing the same code from scratch for multiple projects. Shared libraries share code across development projects, thus optimizing the **software development life cycle**. This drastically cuts down time spent on coding and helps avoid duplicate code.

- Creating a shared library also simplifies the process of pushing source code updates for a project. Updating the library source code also updates the code of every project that uses the library.

→Create a new Git repository containing a directory named *vars*. Save the script in the *vars* directory as *hello.groovy*.



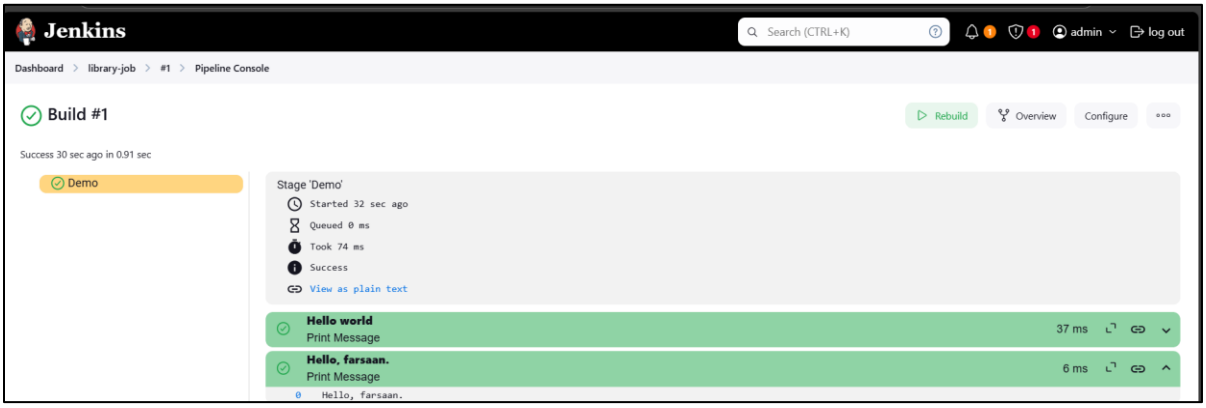
→Add a Shared Library in Jenkins

Dashboard > manage jenkins> system > Global Trusted Pipeline Libraries > name > version > scm > project repository > save

→CREATE A NEW JENKINS JOB WITH PIPELINE AND ADD THE BELOW SCRIPT

```
@Library('git-repo-library@main') _

pipeline {
  agent any
  stages {
    stage('Demo') {
      steps {
        echo 'Hello world'
        hello()  // Call the shared library function without providing a name
      }
    }
  }
}
```

→BUILD



9) Create One Jenkins job to Build and push the Docker image to Docker Hub. (https://github.com/betawins/Python-app.git)

Prerequisites

1. **Jenkins with Docker installed**: Ensure Jenkins can access Docker.
2. **Docker Hub Credentials**: Add your Docker Hub credentials in Jenkins

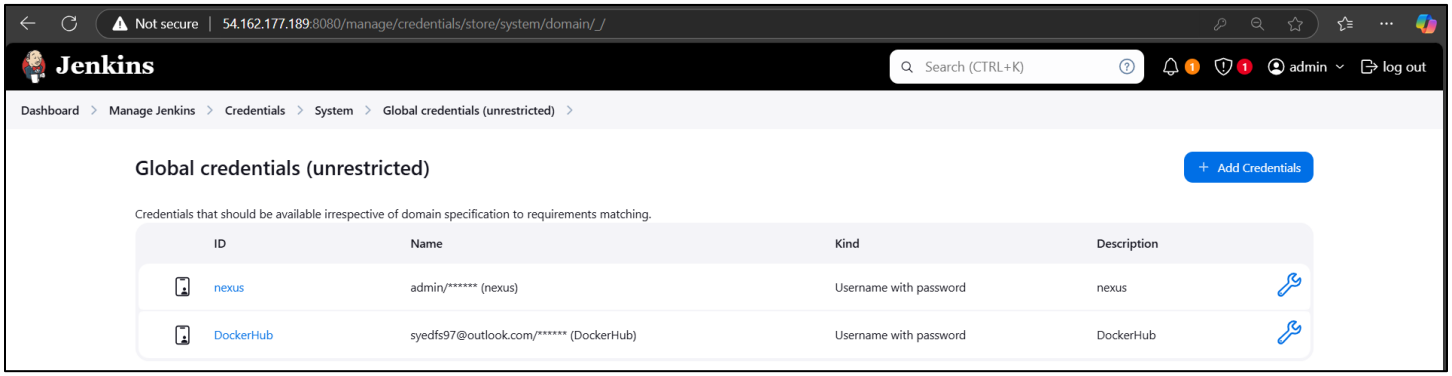
→INSTALLING DOCKER ON EXISTING JENKINS SERVER

```
# sudo yum install docker -y
# sudo service docker start
# sudo usermod -aG docker jenkins
# sudo systemctl restart jenkins
# sudo systemctl enable docker
# docker --version
# systemctl status docker
```

→CHECK THE SERVICE

```
[ec2-user@ip-10-0-0-17 ~]$ docker --version
Docker version 25.0.6, build 32b99dd
[ec2-user@ip-10-0-0-17 ~]$ ssytemctl status docker
-bash: ssytemctl: command not found
[ec2-user@ip-10-0-0-17 ~]$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: https://docs.docker.com
[ec2-user@ip-10-0-0-17 ~]$ systemctl start docker
Failed to start docker.service: The name org.freedesktop.PolicyKit1 was not provided by any .service files
See system logs and 'systemctl status docker.service' for details.
[ec2-user@ip-10-0-0-17 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-10-0-0-17 ~]$ docker --version
Docker version 25.0.6, build 32b99dd
[ec2-user@ip-10-0-0-17 ~]$ systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since Wed 2024-12-25 20:18:58 UTC; 2min 28s ago
```

→ADD CREDENTIALS IN JENKINS GUI



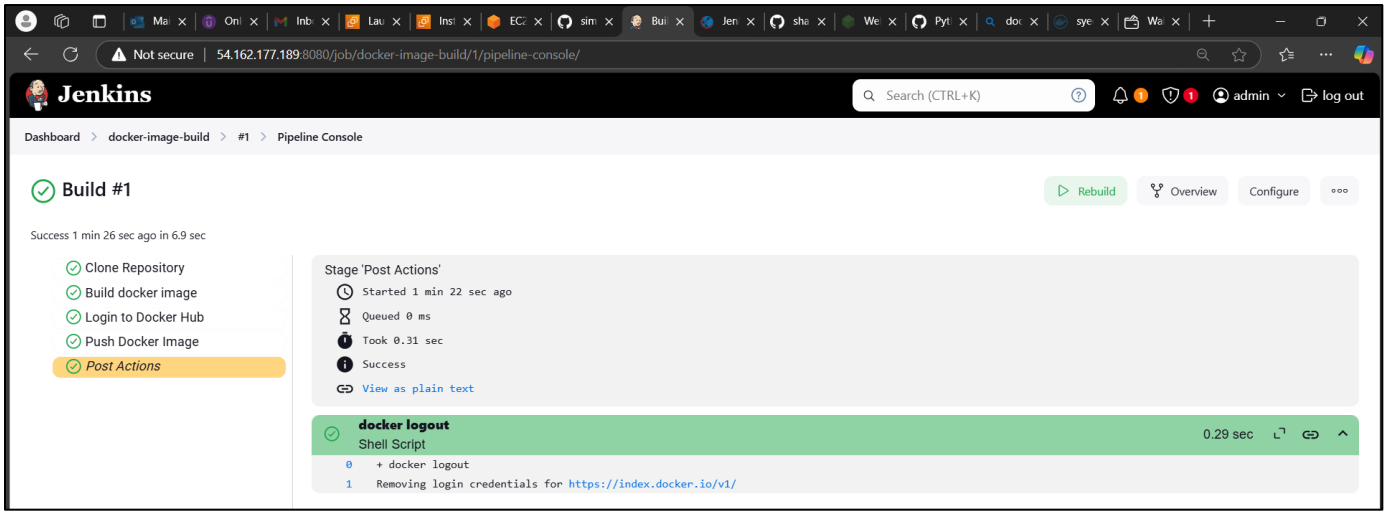
<MAKE SURE YOU CREATE A REPOSITORY IN DOCKERHUB ACCOUNT OF MENTIONED THE CREDENTIALS AND YOU ARE SPECIFYING SAME REPO IN PIPELINE>

Create a jenkins job

Dashboard > new item > item name > select pipeline > add below script > save

```
pipeline {
  agent any
  environment {
    DOCKERHUB_CREDENTIALS = credentials('DockerHub')
  }
  stages {
    stage('Clone Repository') {
      steps {
        // Clone the repository
        git url: 'https://github.com/betawins/Python-app.git', branch: 'main'

        // List files to verify the Dockerfile is present
        sh 'ls -al'
      }
    }
    stage('Build docker image') {
      steps {
        sh 'docker build -t syedfs97/nginx-repo:$BUILD_NUMBER .'
      }
    }
    stage('Login to Docker Hub') {
      steps {
        sh 'echo $DOCKERHUB_CREDENTIALS_PSW | docker login -u $DOCKERHUB_CREDENTIALS_USR --password-stdin'
      }
    }
    stage('Push Docker Image') {
      steps {
        sh 'docker push syedfs97/nginx-repo:$BUILD_NUMBER'
      }
    }
  }
  post {
    always {
      sh 'docker logout'
    }
  }
}
```

→Build and check the pipeline console



CHECKING THE DOCKERHUB REPOSITORY