

# 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](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

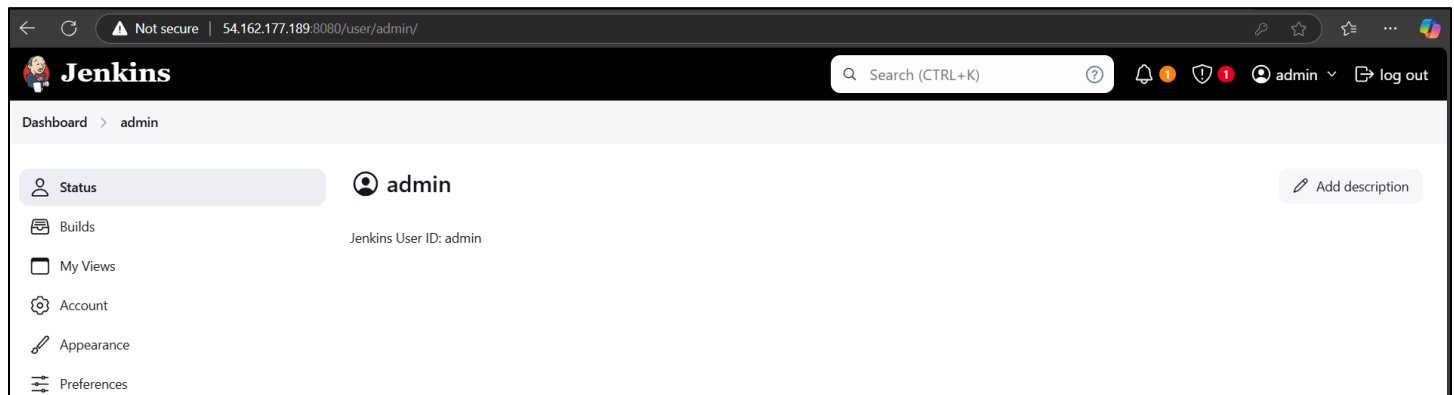
# Open port 8080 for Jenkins
sudo firewall-cmd --permanent --zone=public --add-port=8080/tcp
sudo firewall-cmd --reload
```

→ 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

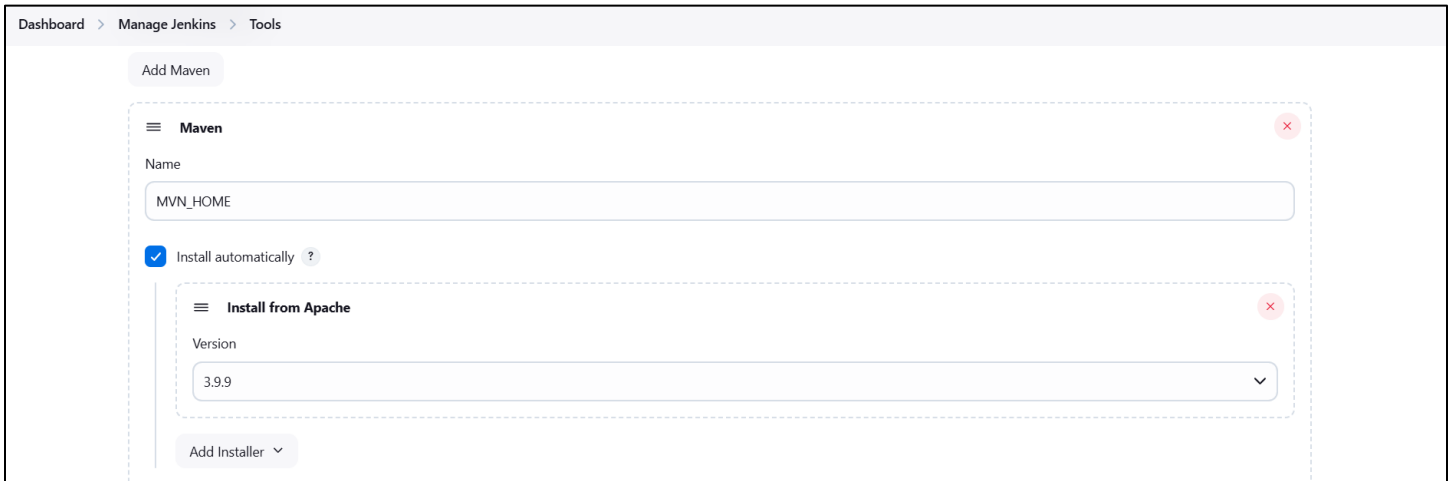
settings > repositories > create repository > name > configurations > create



<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.



The screenshot shows the Jenkins 'Global Tool Configuration' page for Maven. The breadcrumb trail at the top is 'Dashboard > Manage Jenkins > Tools'. There is an 'Add Maven' button at the top left. The main configuration area is titled 'Maven' and contains a 'Name' field with the value 'MVN\_HOME'. Below this is a checked checkbox for 'Install automatically'. Underneath is a section titled 'Install from Apache' with a 'Version' dropdown menu set to '3.9.9'. At the bottom left of the configuration area is an 'Add Installer' button with a dropdown arrow. A red 'X' icon is visible in the top right corner of the configuration box.

- **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.

Global credentials (unrestricted)				<a href="#">+ Add Credentials</a>
Credentials that should be available irrespective of domain specification to requirements matching.				
ID	Name	Kind	Description	
 nexus	admin/***** (nexus)	Username with password	nexus	

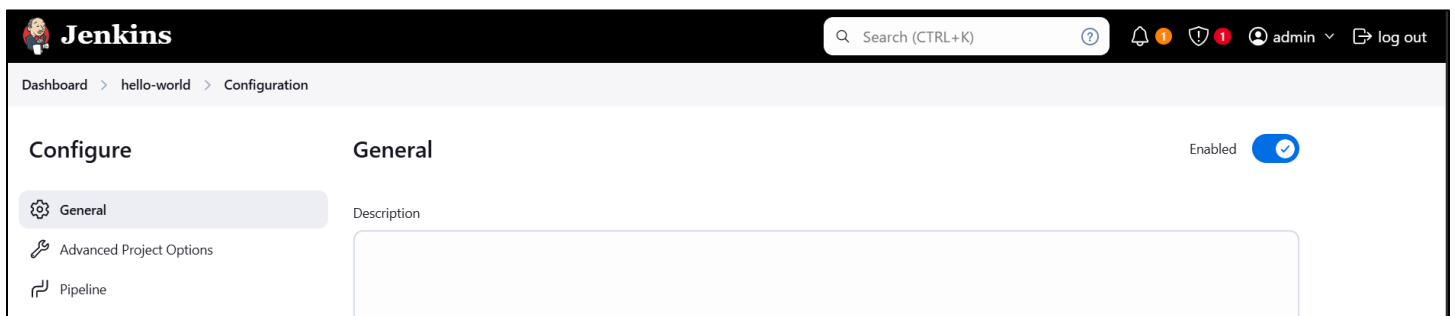
### UPDATED SCRIPT FOR DECLARATIVE PIPELINE

```
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 {
```

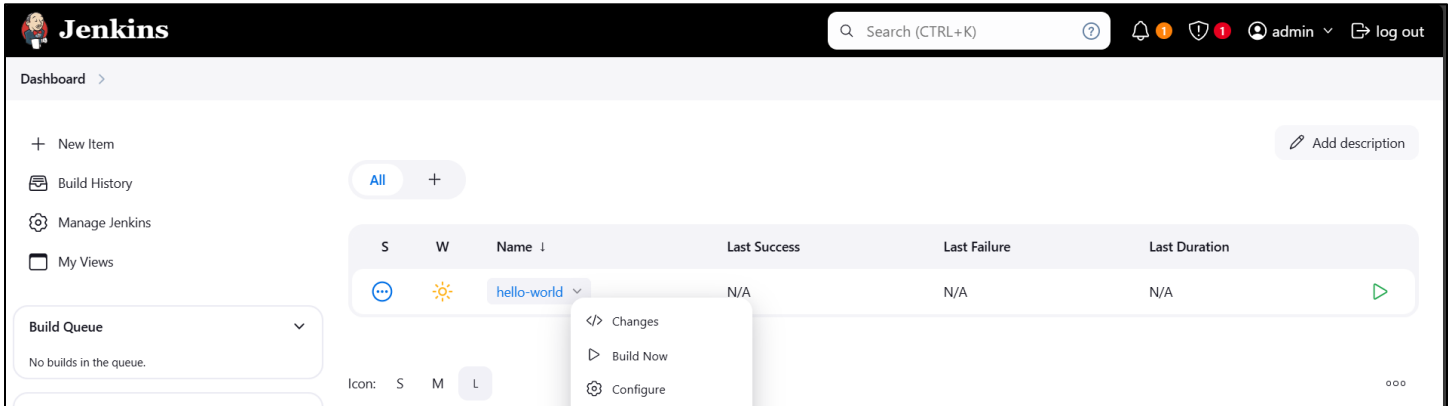
```
sh 'mvn -Dmaven.test.failure.ignore=true install'
}
}
stage("Publish to Nexus") {
    steps {
        script {
            // Manually set the path of the artifact based on the packaging type
            def artifactPath = "target/${env.PACKAGING}-artifact.${env.PACKAGING}"
            // Check if 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

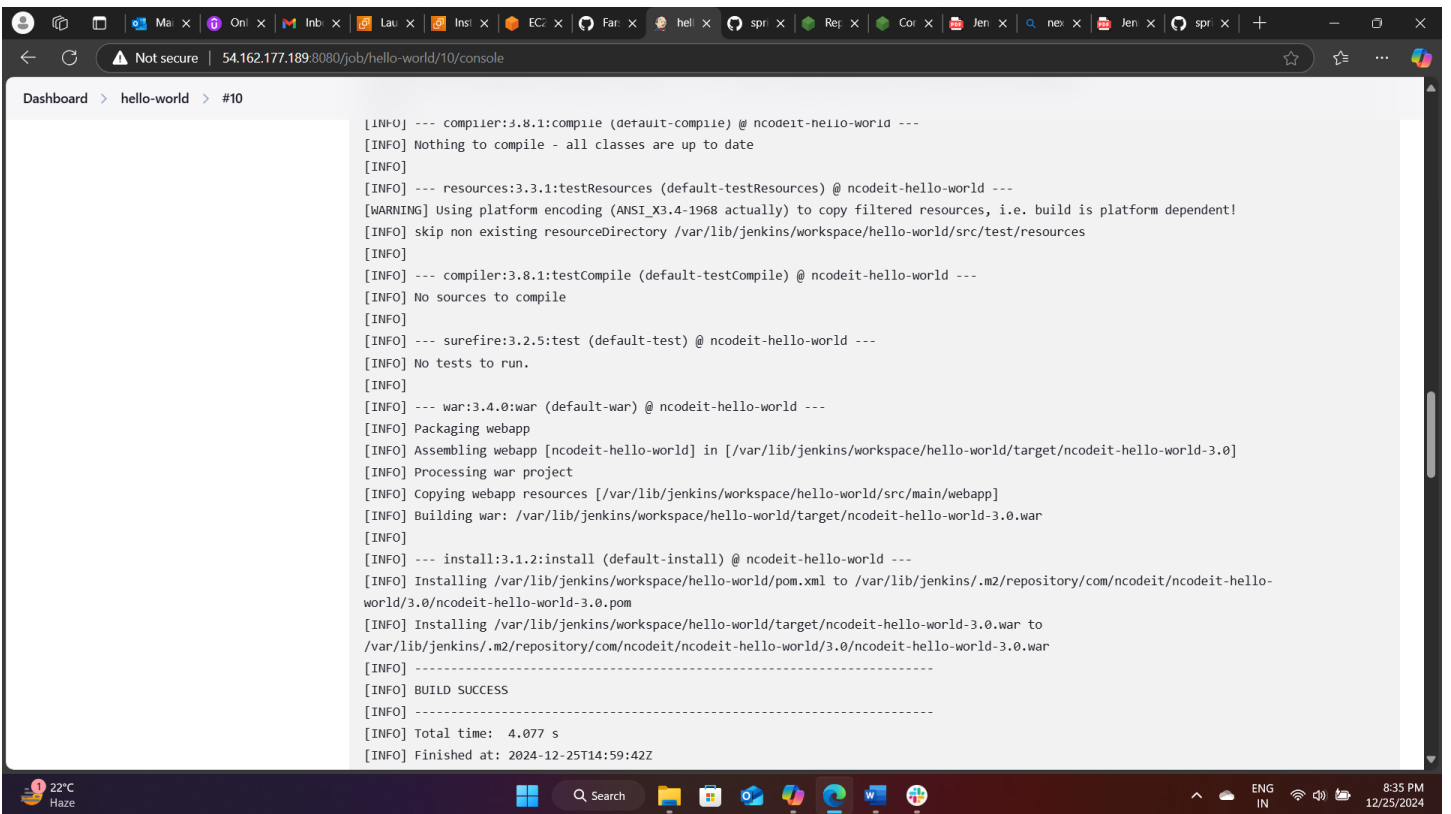


Pipeline > pipeline script from SCM > SCM > git > Repository URL > <https://github.com/Farsaan-tech/spring3-mvc-maven-xml-hello-world-1.git> > Branch > Master > script path > Jenkinsfile > Save



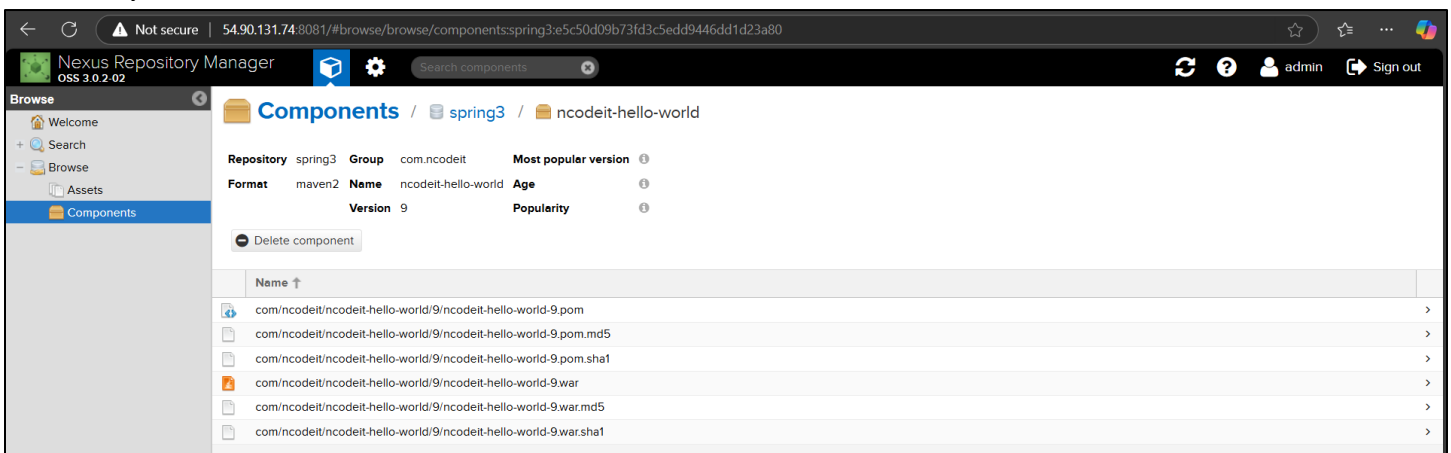
The screenshot shows the Jenkins Dashboard. On the left, there's a sidebar with 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below this is a 'Build Queue' section stating 'No builds in the queue.' The main area displays a table of builds. The first build is named 'hello-world' with a status of 'Success' (green sun icon). A dropdown menu is open for this build, showing options: 'Changes', 'Build Now', and 'Configure'. The table headers are 'S', 'W', 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. The 'hello-world' build has 'N/A' for both 'Last Success' and 'Last Failure', and 'N/A' for 'Last Duration'. There are also icons for 'Icon: S', 'M', and 'L'.

## → Build Now



The screenshot shows the Jenkins Console Output for the 'hello-world' build. The output is a log of Maven commands and their results. It starts with '[INFO] --- compiler:3.8.1:compile (default-compile) @ ncodeit-hello-world ---' and ends with '[INFO] Finished at: 2024-12-25T14:59:42Z'. The log includes various status messages like '[INFO] Nothing to compile - all classes are up to date', '[WARNING] Using platform encoding (ANSI\_X3.4-1968 actually) to copy filtered resources, i.e. build is platform dependent!', and '[INFO] BUILD SUCCESS'. The console output is displayed in a monospaced font on a light background.

## → Verify on Nexus



The screenshot shows the Nexus Repository Manager interface. The top bar includes the 'Nexus Repository Manager' logo, a search bar, and user information. The main area is titled 'Components' and shows a list of components for the 'spring3' repository. The components are listed in a table with columns: 'Repository', 'Group', 'Name', 'Version', 'Most popular version', 'Age', and 'Popularity'. The components are 'com.ncodeit:ncodeit-hello-world-9-pom', 'com.ncodeit:ncodeit-hello-world-9-pom.md5', 'com.ncodeit:ncodeit-hello-world-9-pom.sha1', 'com.ncodeit:ncodeit-hello-world-9-war', 'com.ncodeit:ncodeit-hello-world-9-war.md5', and 'com.ncodeit:ncodeit-hello-world-9-war.sha1'. A 'Delete component' button is visible below the table.

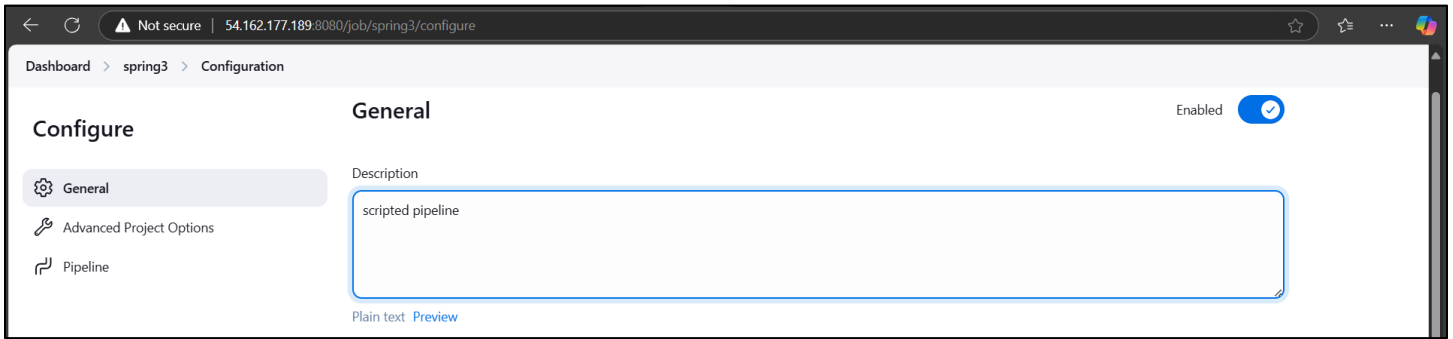
## 2) CREATE ONE SCRIPTED PIPELINE JOB

<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

The Jenkins interface shows the build page for job 'spring3' at build #4. The build is successful, indicated by a green checkmark. It was started by user 'admin' on Dec 25, 2024, at 3:18:27 PM. The build took 8.2 seconds. The console output shows the artifact upload process. The build is linked to a Git repository: <https://github.com/Farsaan-tech/spring3-mvc-maven-xml-hello-world-1.git>, revision 094a103f8203ad906116fe95dcbbfccaa6d528a8. The build is configured to keep this build forever.

## → Verify on Nexus

The Nexus Repository Manager interface shows the 'Assets' view for the 'spring-scripted' repository. The assets are listed as follows:

Name	Size	Checksums
com/ncodeit/ncodeit-hello-world/4/ncodeit-hello-world-4.pom	1.5 KB	MD5, SHA1
com/ncodeit/ncodeit-hello-world/4/ncodeit-hello-world-4.pom.md5	1.5 KB	MD5, SHA1
com/ncodeit/ncodeit-hello-world/4/ncodeit-hello-world-4.pom.sha1	1.5 KB	MD5, SHA1
com/ncodeit/ncodeit-hello-world/4/ncodeit-hello-world-4.war	1.5 KB	MD5, SHA1
com/ncodeit/ncodeit-hello-world/4/ncodeit-hello-world-4.war.md5	1.5 KB	MD5, SHA1
com/ncodeit/ncodeit-hello-world/4/ncodeit-hello-world-4.war.sha1	1.5 KB	MD5, SHA1



### 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,
```



#### 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...'
      }
    }
  }
}
```

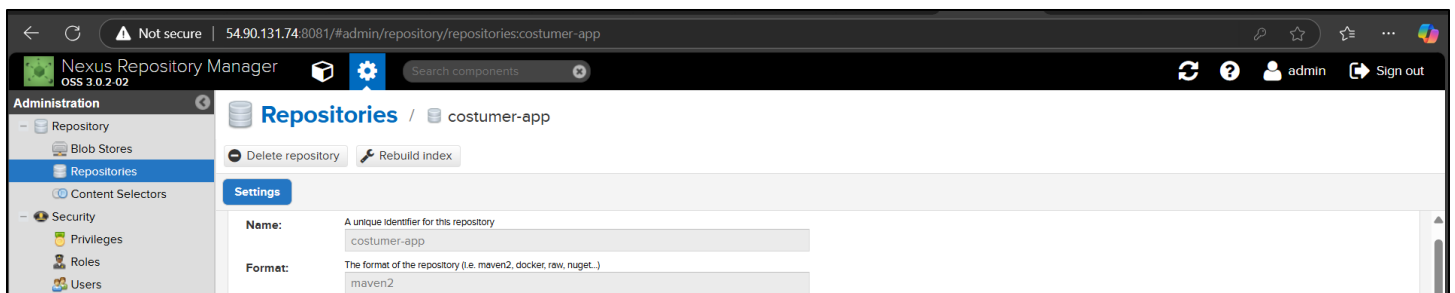
```
node {
  stage('Build') {
    echo 'Building...'
  }
  stage('Test') {
    echo 'Testing...'
  }
  stage('Deploy') {
    echo 'Deploying...'
  }
  // Post actions
  always {
    echo 'This will always run'
  }
  success {
    echo 'This will run only if successful'
  }
  failure {
    echo 'This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

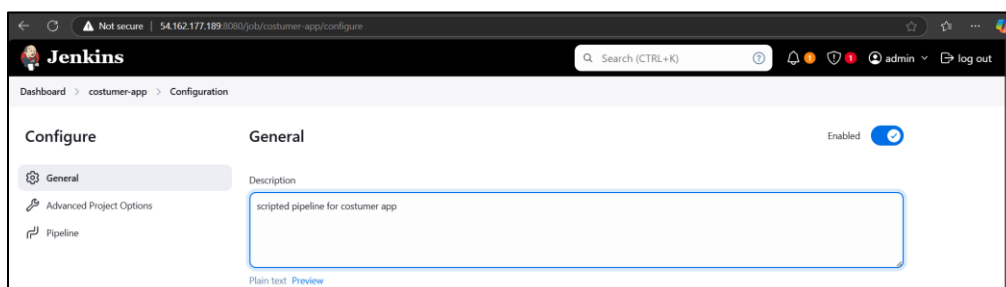
```

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"
        }
      }
    }
  }
}
}
}
}

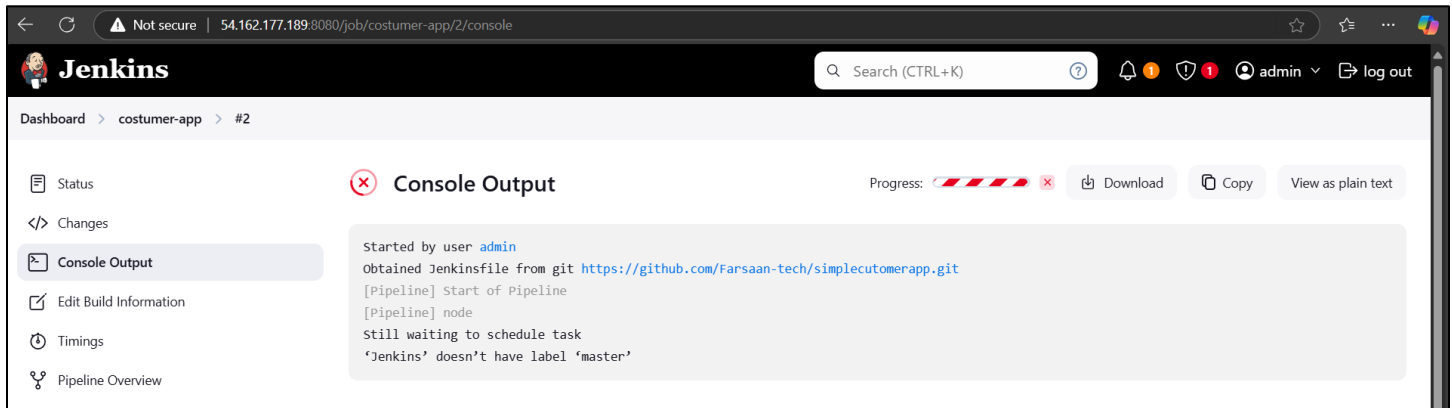
```

## Create a jenkins job

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

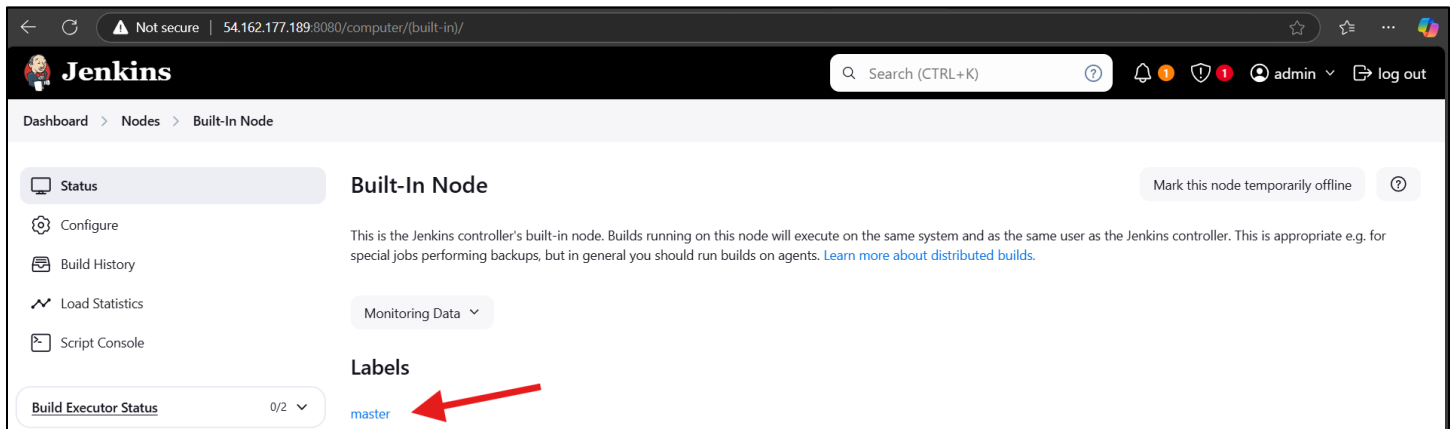


Pipeline > pipeline script from SCM > SCM > git > Repository URL > <https://github.com/Farsaan-tech/simplecutoomerapp.git> > Branch > master > script path > Jenkinsfile > Save > Build Now

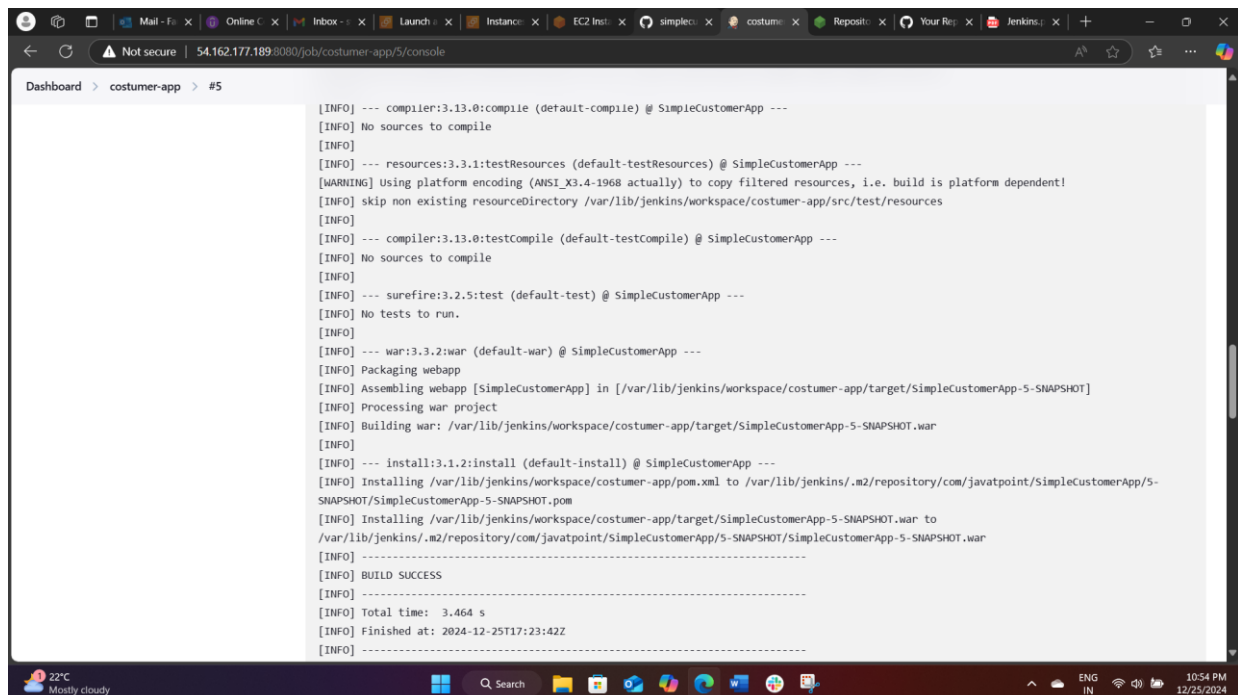


## → ADD A NODE WITH LABEL MASTER

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>

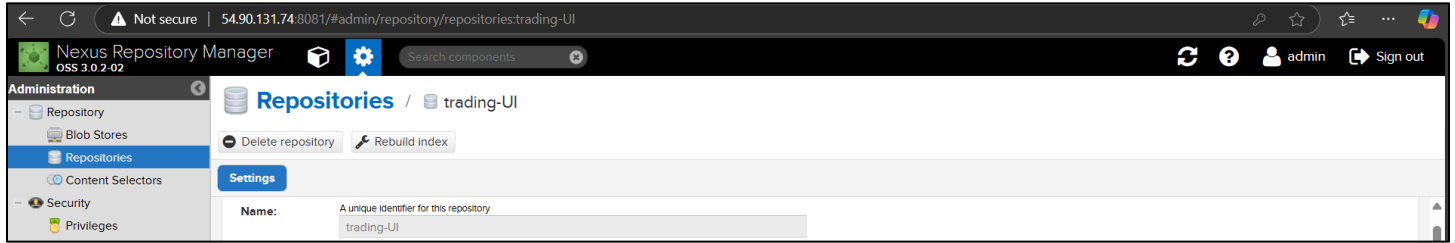
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <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

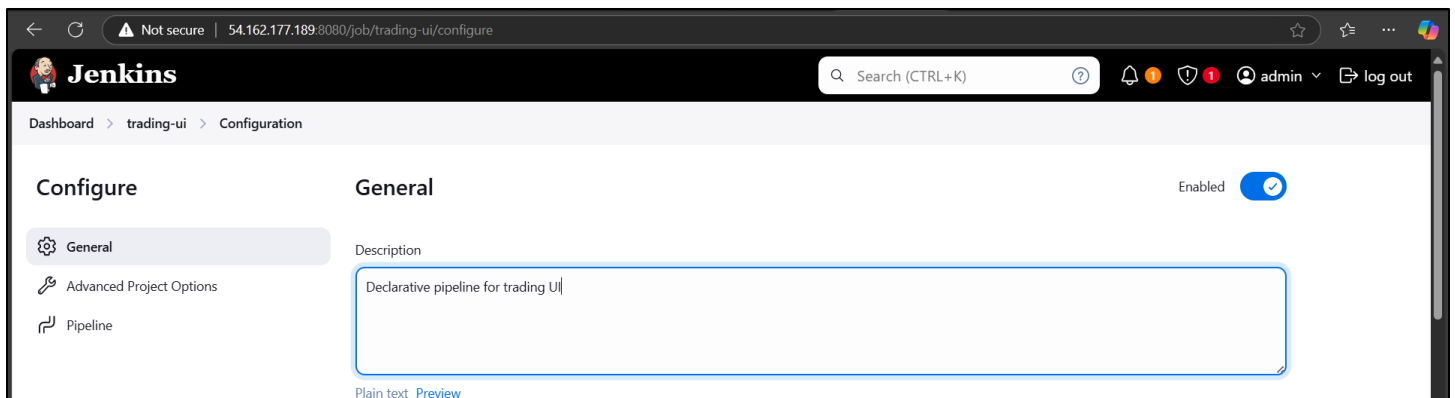
→GIT CLONE

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



## → BUILD NOW

The Jenkins interface displays the status of build #9 for the 'trading-ui' job. The build is successful, indicated by a green checkmark. The build started on December 25, 2024, at 6:23:09 PM. The build duration was 9.1 seconds, and the total time from scheduled to completion was 9.1 seconds. The build was started by the user 'admin'. The build output shows the revision 0a07db1e9e271e97da6733dd6fad26e3ff713ea5 from the repository https://github.com/Farsaan-tech/Trading-UI.git. The build steps show 'No changes'.

**Jenkins** Search (CTRL+K) admin log out

Dashboard > trading-ui > #9

**Status** #9 (Dec 25, 2024, 6:23:09 PM) Add description Keep this build forever

Changes

Console Output

Edit Build Information

Delete build '#9'

Timings

Git Build Data

Pipeline Overview

Pipeline Console

Restart from Stage

Replay

Pipeline Steps

Workspaces

Previous Build

Started by user admin

This run spent:

- 17 ms waiting;
- 9.1 sec build duration;
- 9.1 sec total from scheduled to completion.

Revision: 0a07db1e9e271e97da6733dd6fad26e3ff713ea5  
Repository: https://github.com/Farsaan-tech/Trading-UI.git

refs/remotes/origin/patch-1

No changes.

AUS - IND In 5 hours 11:58 PM 12/25/2024

## → NEXUS ARTIFACTORY

The Nexus Repository Manager interface shows the 'Assets' view for the 'trading-ui' repository. The assets are listed in a table with columns for Name and a download icon. The assets are:

- com/example/trading-ui/9/trading-ui-9.pom
- com/example/trading-ui/9/trading-ui-9.pom.md5
- com/example/trading-ui/9/trading-ui-9.pom.sha1
- com/example/trading-ui/9/trading-ui-9.war
- com/example/trading-ui/9/trading-ui-9.war.md5
- com/example/trading-ui/9/trading-ui-9.war.sha1

The interface also includes a search bar, a filter input, and a sidebar with navigation options like Welcome, Search, Browse, Assets, and Components.

Nexus Repository Manager OSS 3.0.2-02

Assets / trading-ui

Filter

Name ↑	
com/example/trading-ui/9/trading-ui-9.pom	>
com/example/trading-ui/9/trading-ui-9.pom.md5	>
com/example/trading-ui/9/trading-ui-9.pom.sha1	>
com/example/trading-ui/9/trading-ui-9.war	>
com/example/trading-ui/9/trading-ui-9.war.md5	>
com/example/trading-ui/9/trading-ui-9.war.sha1	>

Copyright © 2008-present, Sonatype Inc. All rights reserved.

21°C Partly cloudy 12:00 AM 12/26/2024

## 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

The screenshot shows the Jenkins web interface. At the top, the browser address bar displays '54.162.177.189:8080/job/parrelel-job/2/pipeline-graph/'. The Jenkins header includes a search bar and user information. The breadcrumb trail is 'Dashboard > parrelel-job > #2 > Pipeline Overview'. The main section is titled 'Build #2' with a green checkmark icon. To the right of the title are buttons for 'Rebuild', 'Console', and 'Configure'. Below the title, a 'Pipeline' graph is shown, illustrating the workflow: 'Start' -> 'Parallel Tasks' (which branches into 'Task 1', 'Task 2', and 'Task 3') -> 'Final Task' -> 'End'. All stages and tasks are marked with green checkmarks, indicating successful completion. On the right side, a 'Details' panel provides additional information: 'Manually run by admin', 'Started 1 min 1 sec ago', 'Queued 1 ms', and 'Took 1.3 sec'.

## 8) EXECUTE JENKINS PIPELINE STAGES USING SHARED LIBRARIES

### WHAT IS A SHARED LIBRARY IN JENKINS?

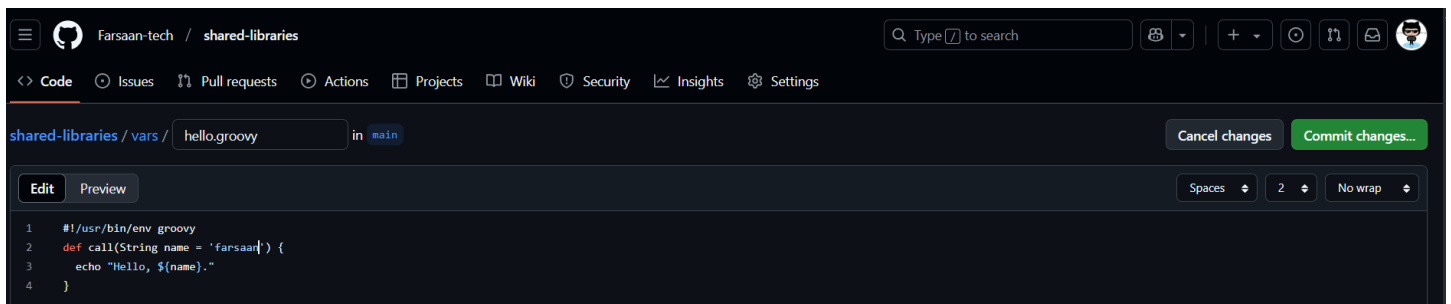
A SHARED LIBRARY IN JENKINS IS A COLLECTION OF GROOVY SCRIPTS SHARED BETWEEN DIFFERENT JENKINS JOBS. TO RUN THE SCRIPTS, THEY ARE PULLED INTO A JENKINSFILE.

### 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

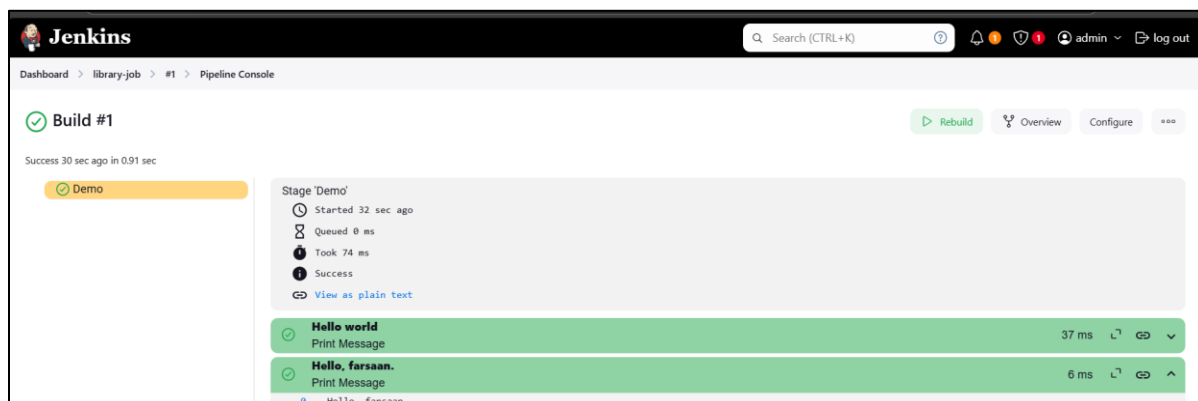
Dashboard > manage jenkins > system > Global Trusted Pipeline Libraries > name > version > scm > project repository > save

→ 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

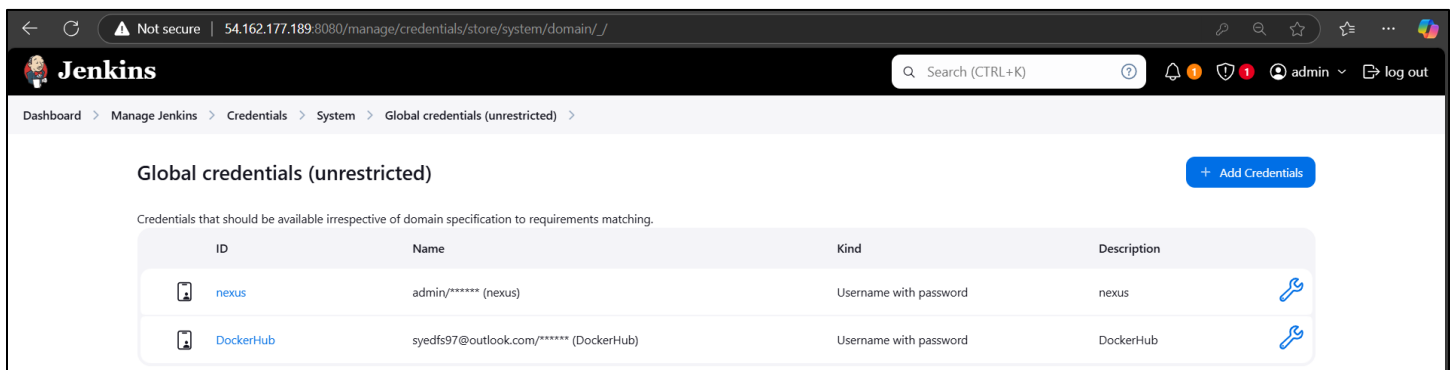
### →INSTALLING DOCKER ON EXISTING JENKINS SERVER

```
# 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 ~]$ ssystemctl status docker
-bash: ssystemctl: 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



The screenshot shows the Jenkins web interface. The breadcrumb navigation is: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted). The page title is "Global credentials (unrestricted)". Below the title is a blue button labeled "+ Add Credentials". A note states: "Credentials that should be available irrespective of domain specification to requirements matching." Below this is a table with the following data:

ID	Name	Kind	Description
nexus	admin/***** (nexus)	Username with password	nexus
DockerHub	syedfs97@outlook.com/***** (DockerHub)	Username with password	DockerHub

<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

The screenshot shows the Jenkins web interface. At the top, the Jenkins logo and a search bar are visible. Below the navigation bar, the breadcrumb trail reads: Dashboard > docker-image-build > #1 > Pipeline Console. The main content area displays 'Build #1' with a green checkmark icon, indicating success. A 'Rebuild' button is present. Below this, a list of stages is shown: 'Clone Repository', 'Build docker image', 'Login to Docker Hub', 'Push Docker Image', and 'Post Actions' (highlighted in orange). The 'Post Actions' stage is expanded, showing a 'docker logout' step with a green checkmark and a duration of 0.29 sec. The console output for this step shows the command 'docker logout' and the message 'Removing login credentials for https://index.docker.io/v1/'.

# CHECKING THE DOCKERHUB REPOSITORY

Ma xOni xInb xLau xInst xEC xsim xBui xJen xsha xWe xPyt xdo xIm xWe x


https://hub.docker.com/repository/docker/syedfs97/nginx-repo/tags/1/sha256-b93657d02d1c40210aa73ccd97084aaf45b4ca576bc91cca8624787d6d7f58fb

dockerhubExploreRepositoriesOrganizationsUsage

Search Docker Hubctrl+K

S

syedfs97 / Repositories / nginx-repo / Tags / 1



## syedfs97/nginx-repo:1

MANIFEST DIGEST sha256:b93657d02d1c40210aa73ccd97084aaf45b4ca576bc91cca8624787d6d7f58fb

Delete Tag

OS/ARCHlinux/amd64

COMPRESSED SIZE333.2 MB

LAST PUSHED3 minutes ago by syedfs97

TYPEImage

MANIFEST DIGESTsha256:b93657d0...

Image Layers

Vulnerabilities

Image Layers

1 ADD file ... in /52.37 MB

2 CMD ["bash"]0 B

3 /bin/sh -c set -eux; apt-get4.91 MB

4 /bin/sh -c set -ex; if10.37 MB

5 /bin/sh -c apt-get update &&52.04 MB

6 /bin/sh -c set -ex; apt-get187.41 MB

Command

ADD file:c03517c5ddbed4053165bdfd984b27a006fb5f533ca80b5798232d96df221440 in /

AUS - INDIn 3 hours

Search

ENG US

2:19 AM12/26/2024