Docker & Kubernetes & SonarQube & Jenkins & Maven are configured on a Single Node

Node Configuration Details – Jenkins & Maven & Docker & Kubernetes

CPU: 8CPU

RAM: 16GB

Hard Disk: 150 GB

IP Address: 10.0.8.45

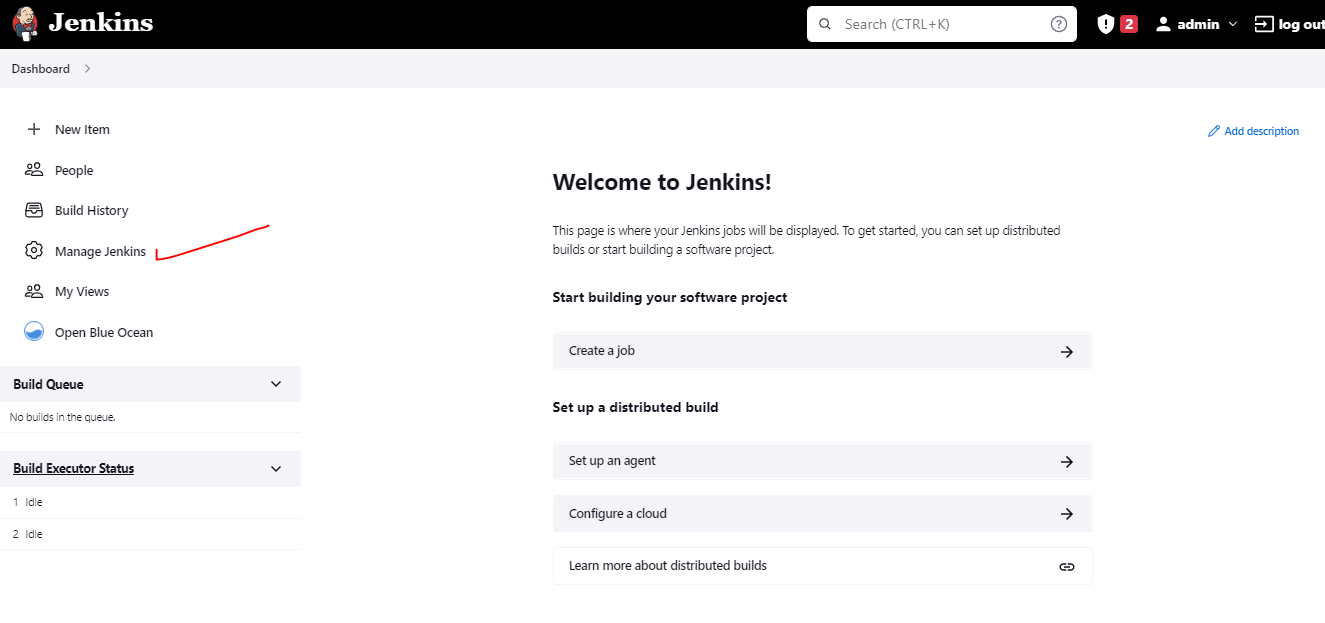
Nexus Configured in a Separate Server

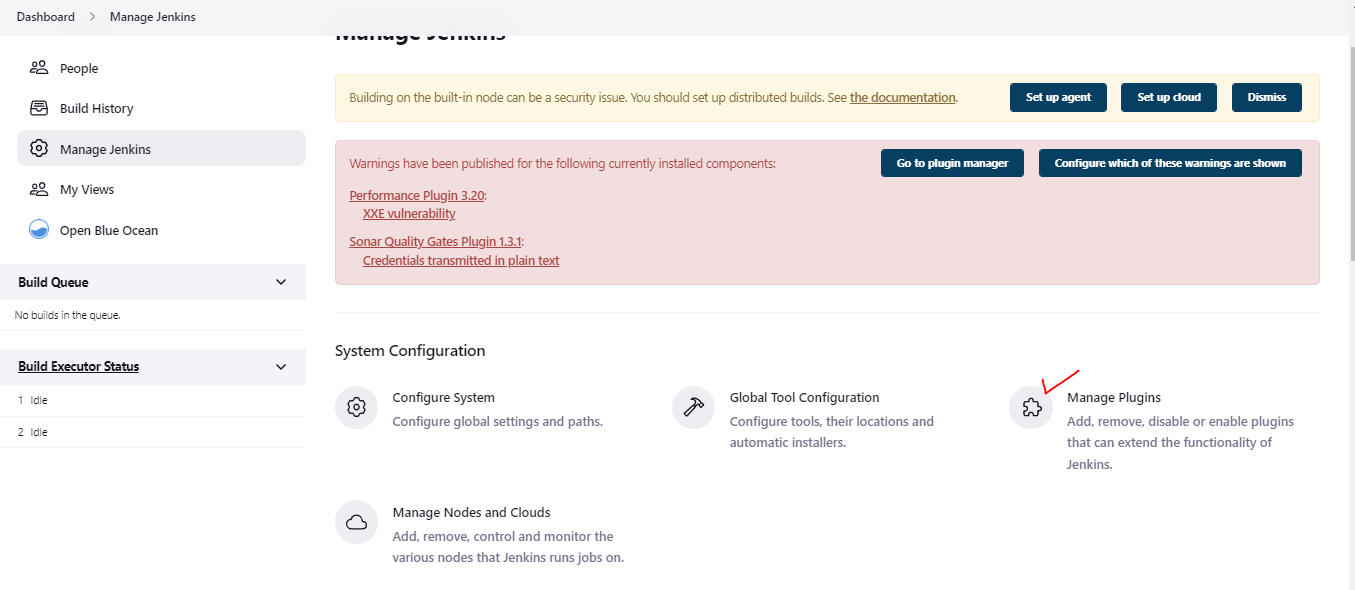
CPU: 4CPU

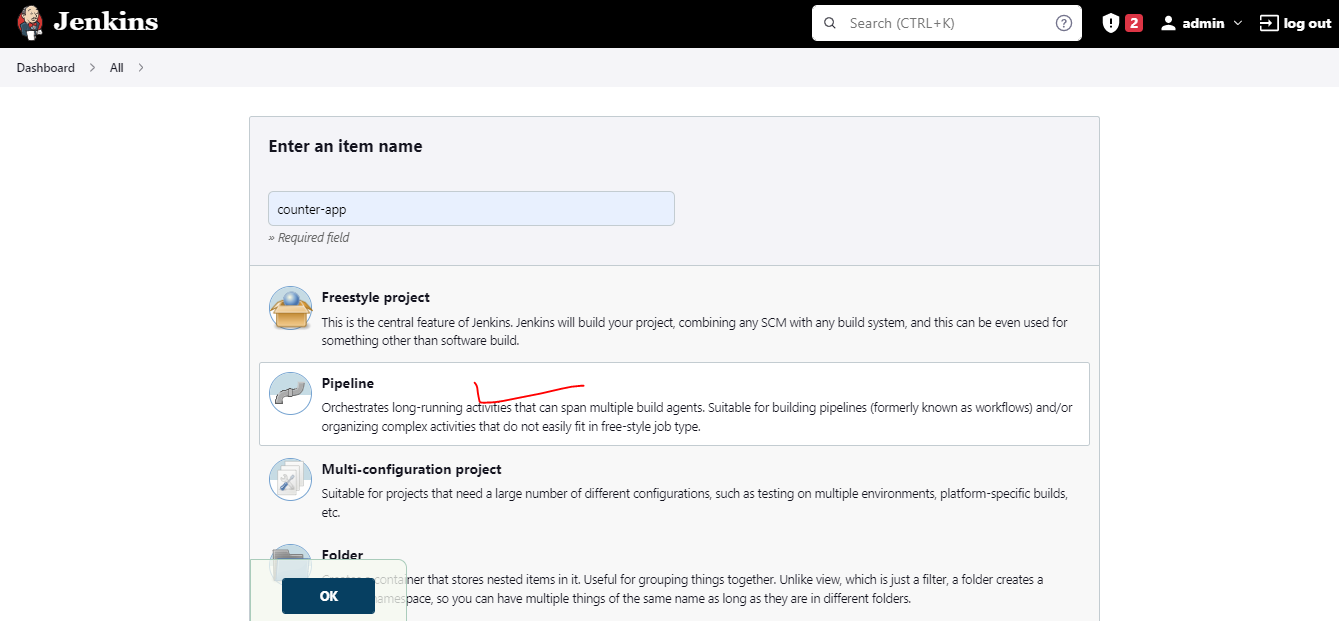
RAM: 16GB

Hard Disk: 60 GB

IP-Address: 10.0.8.74

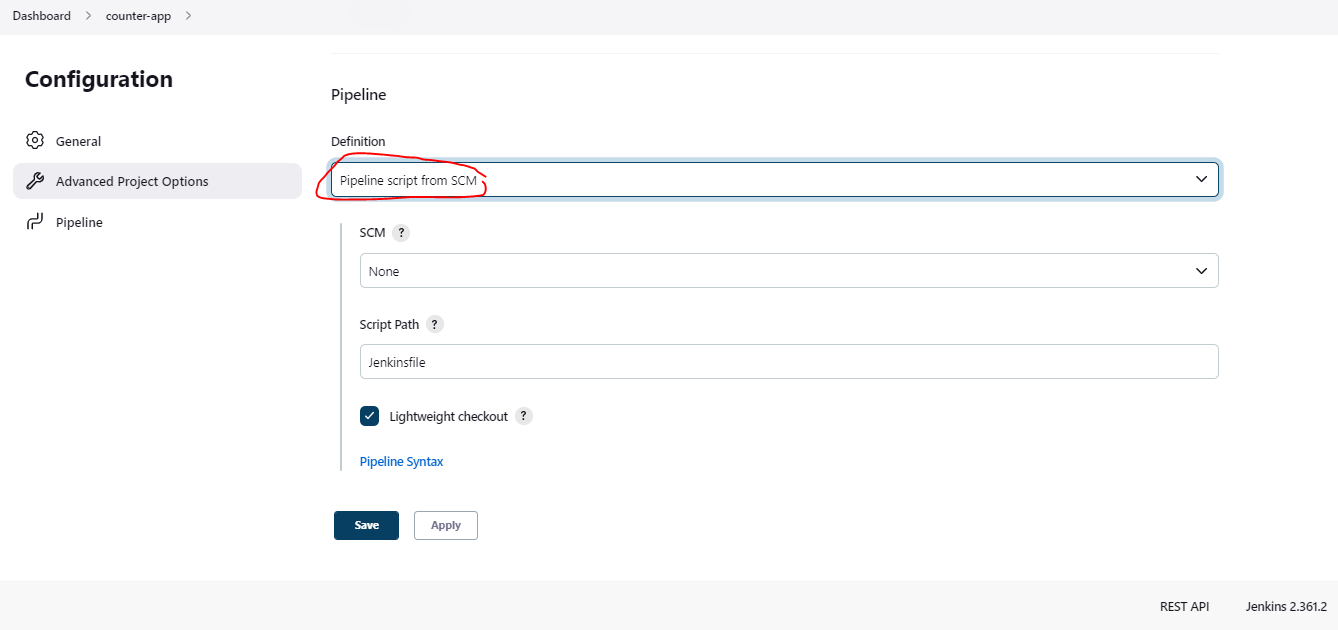


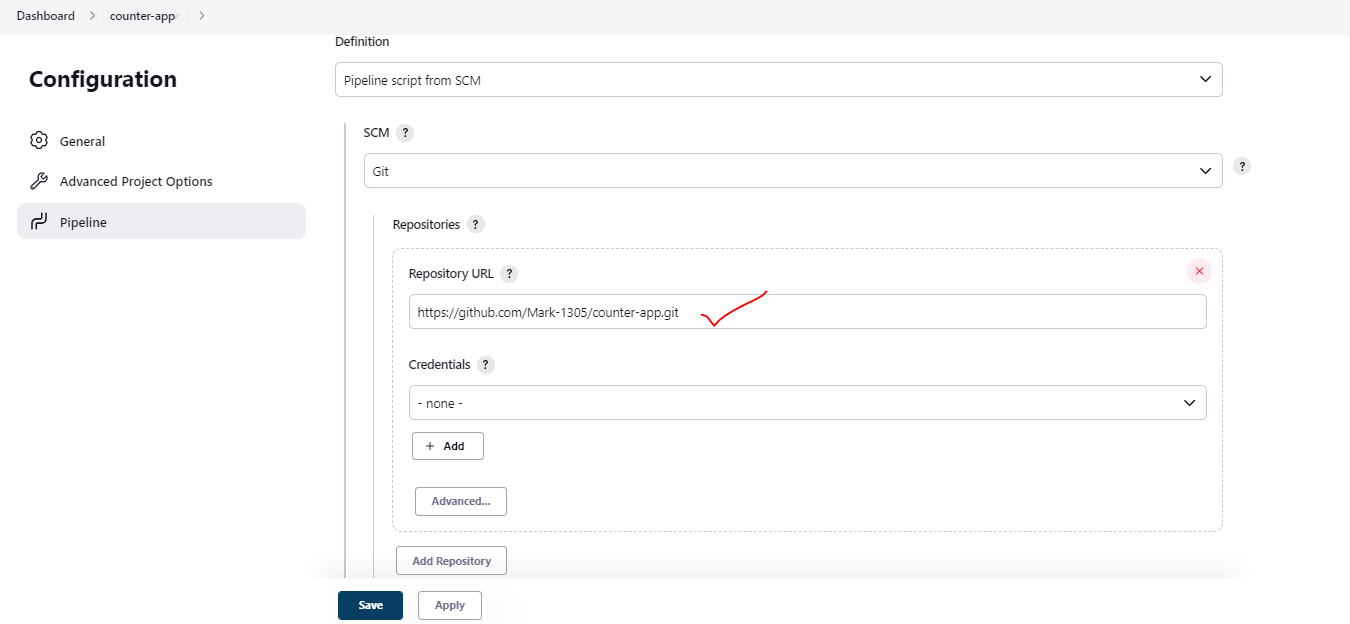


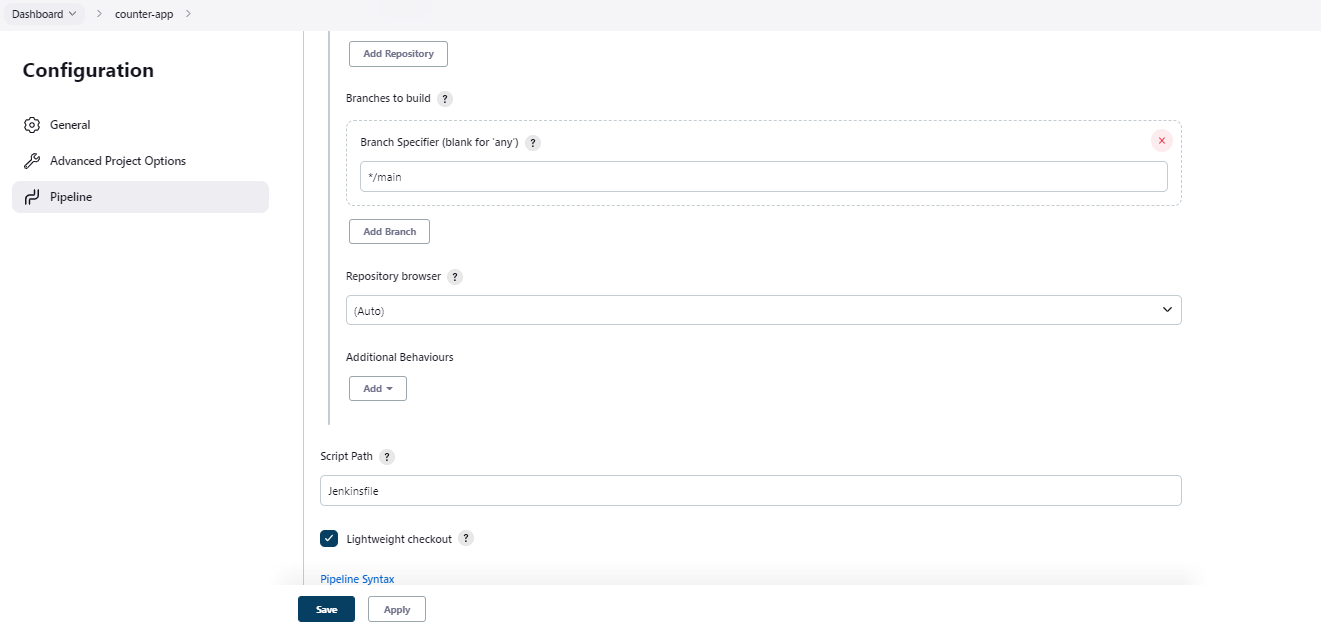


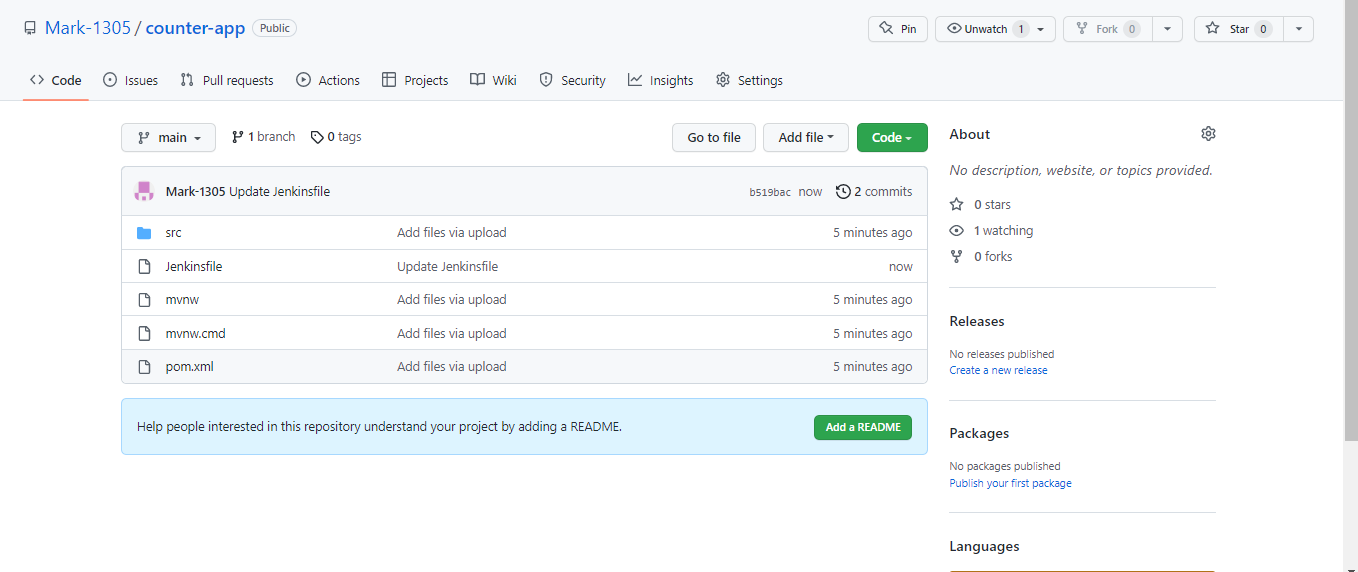
Configure Git Repository to Pull Code:

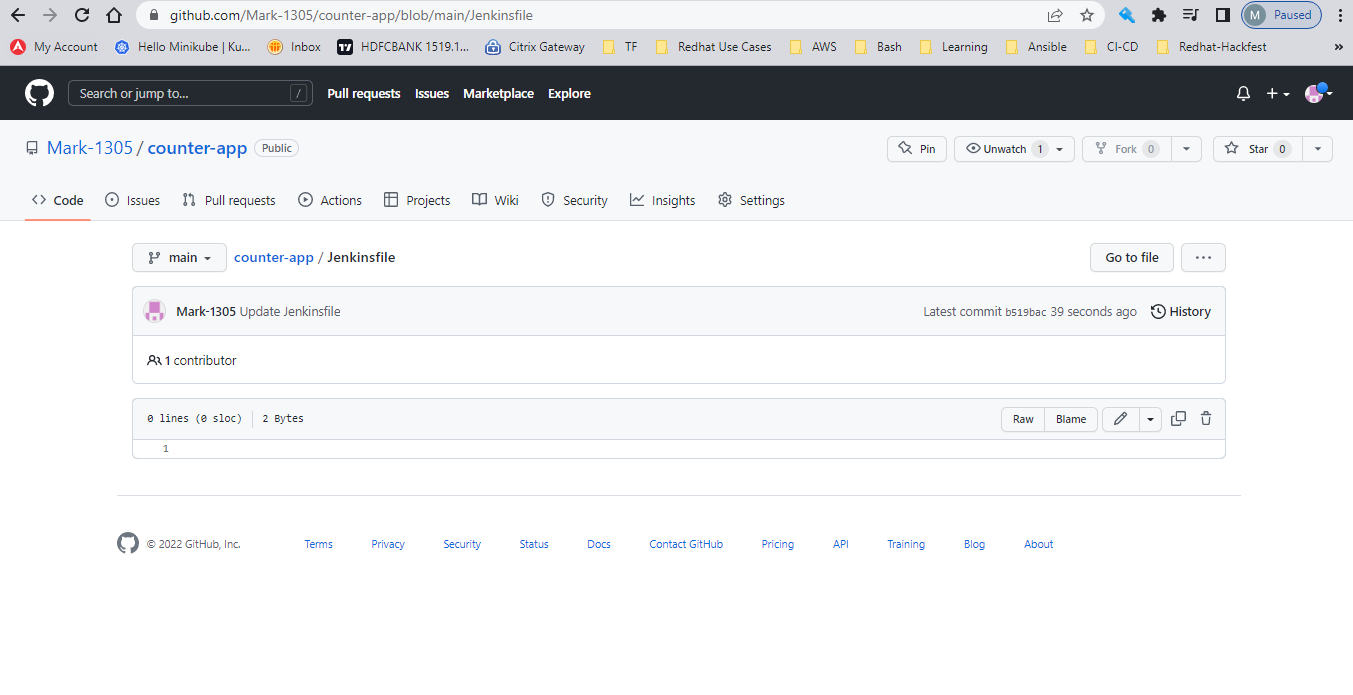
As the repo is public – No Need to Provide any kind of credentials





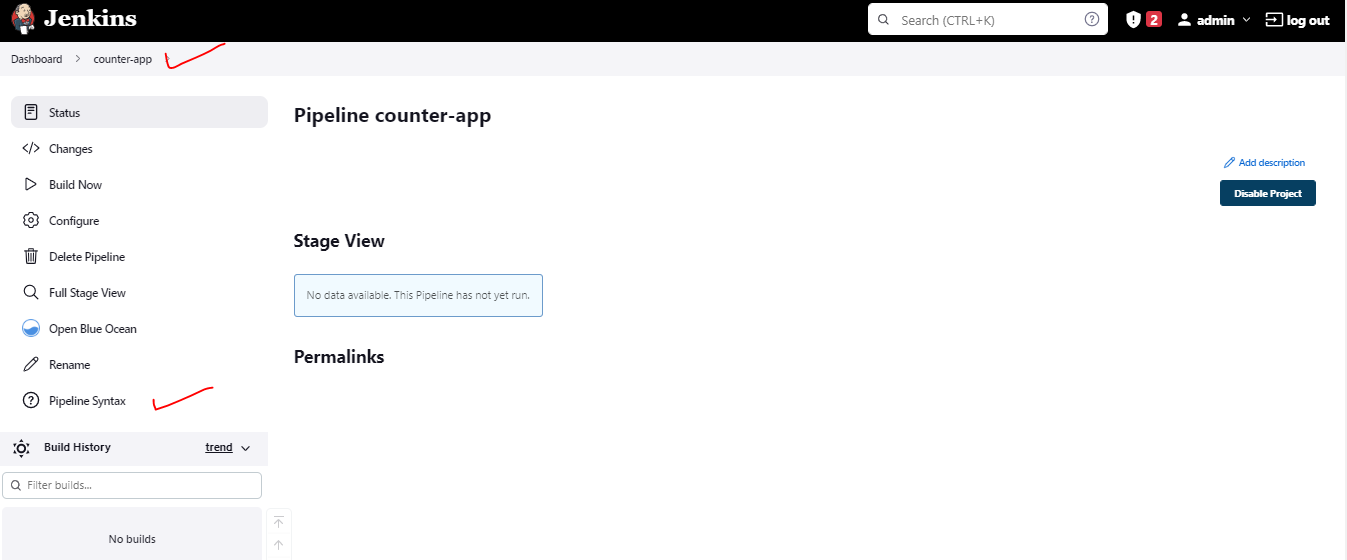


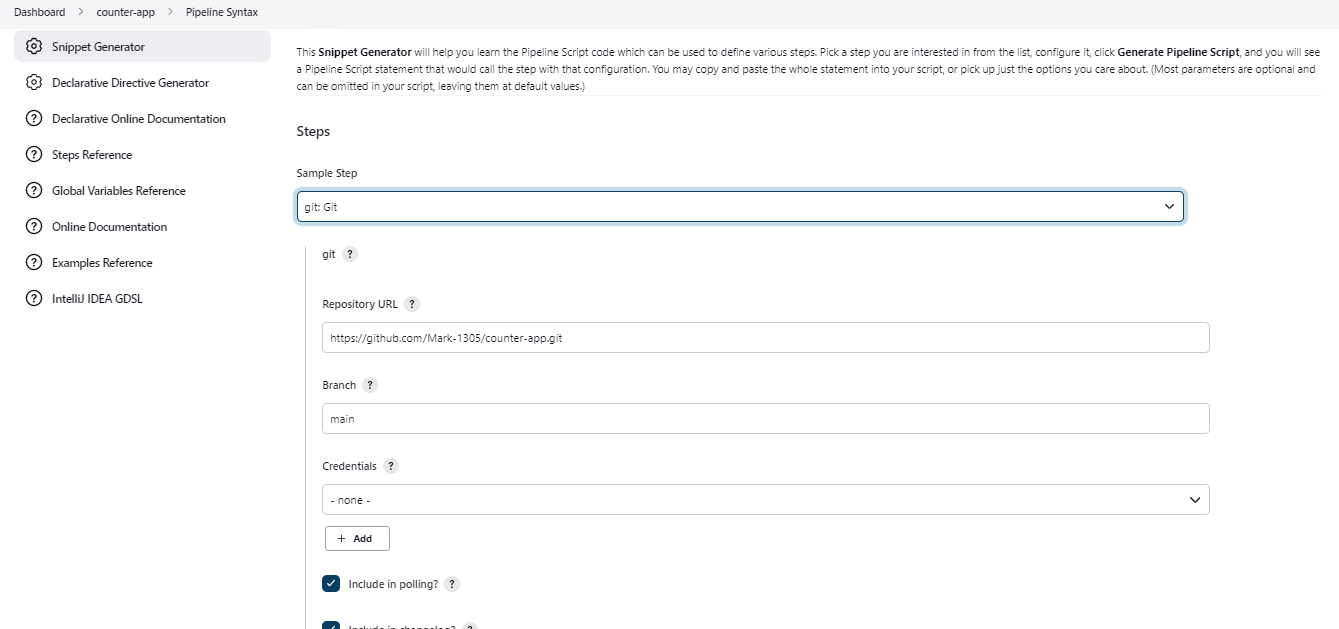




Start Writing Jenkinsfile

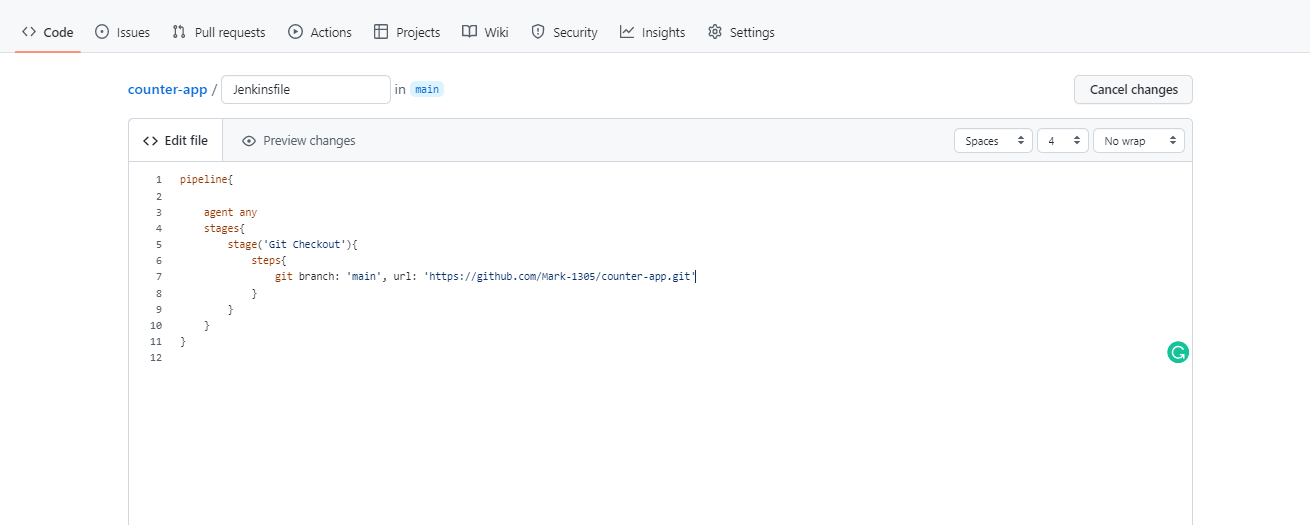
Add Git Repository Syntax – To Run the Job – Use Pipeline syntax





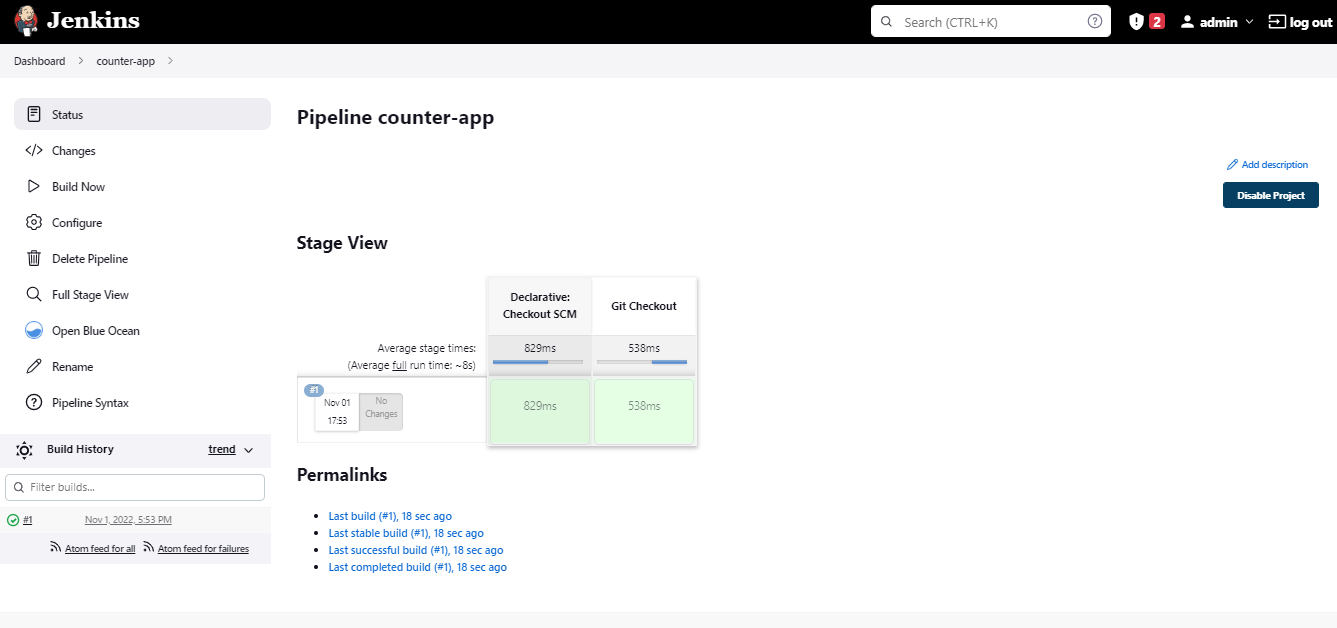
Jenkins file Updated with Git Code generated with Pipeline syntax

**git branch: 'main', url: 'https://github.com/Mark-1305/demo-counter-app.git'**



|  |
| --- |
| pipeline{  agent any  stages{  stage('Git Checkout'){  steps{  git branch: 'main', url: 'https://github.com/Mark-1305/demo-counter-app.git'  }  }  }  } |

Run the Job:

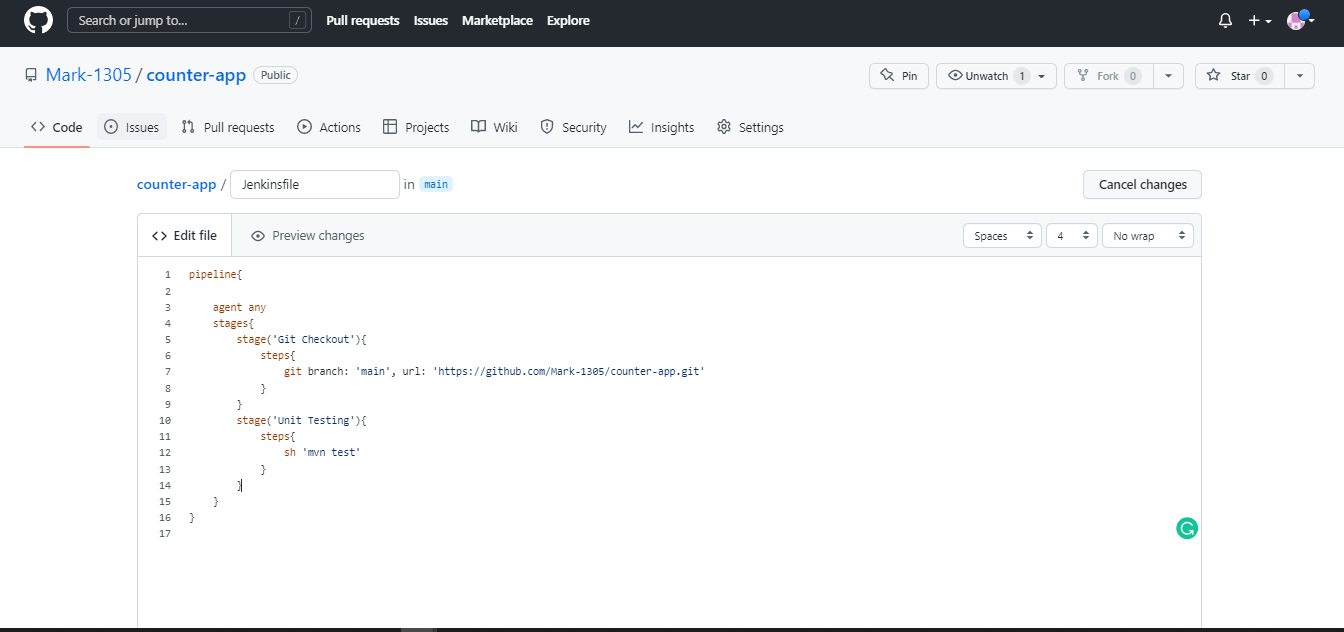


Add second Job

Maven is already Installed in the Local Server

This job is run to do Unit Testing

**Update the code**

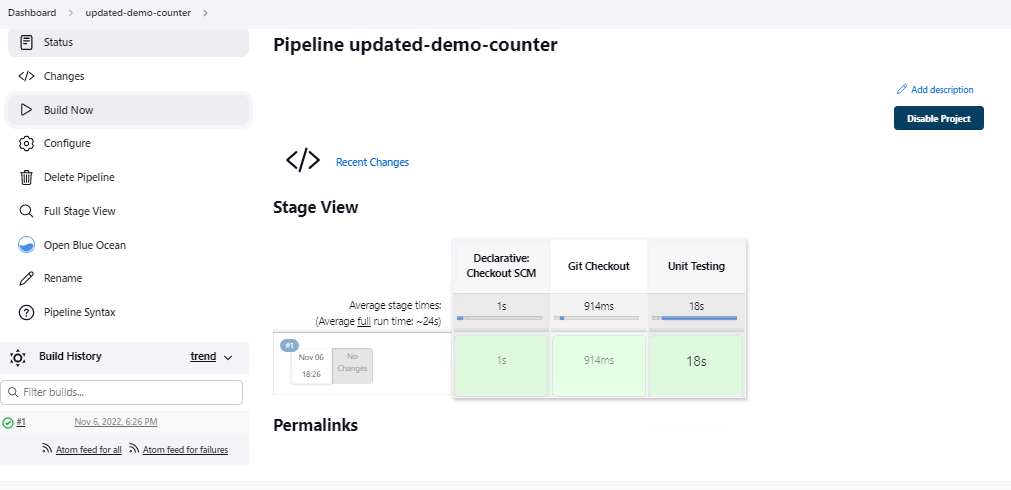


|  |
| --- |
| pipeline{  agent any  stages{  stage('Git Checkout'){  steps{  git branch: 'main', url: 'https://github.com/Mark-1305/demo-counter-app.git'  }  }  stage('Unit Testing'){  steps{  sh 'mvn test'  }  }  }  } |

**Stage: Unit Test**

**Make Sure, Maven is installed locally**

The Unit Test Has been Built – Pipeline name is updated to “updated demo counter”



Let’s Move Forward and do Integration Test

**Stage: Integration Test**

**Add the below code**

**stage('Integration Test'){**

**steps{**

**sh 'mvn verify -DskipUnitTests '**

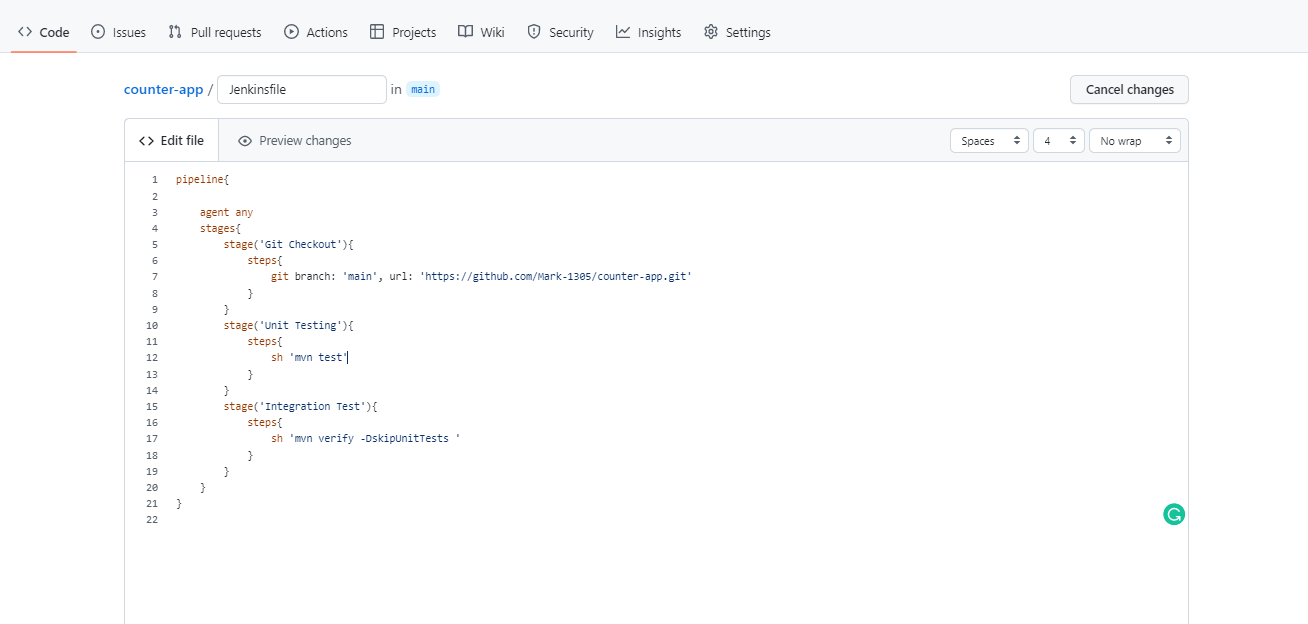
**}**

**}**

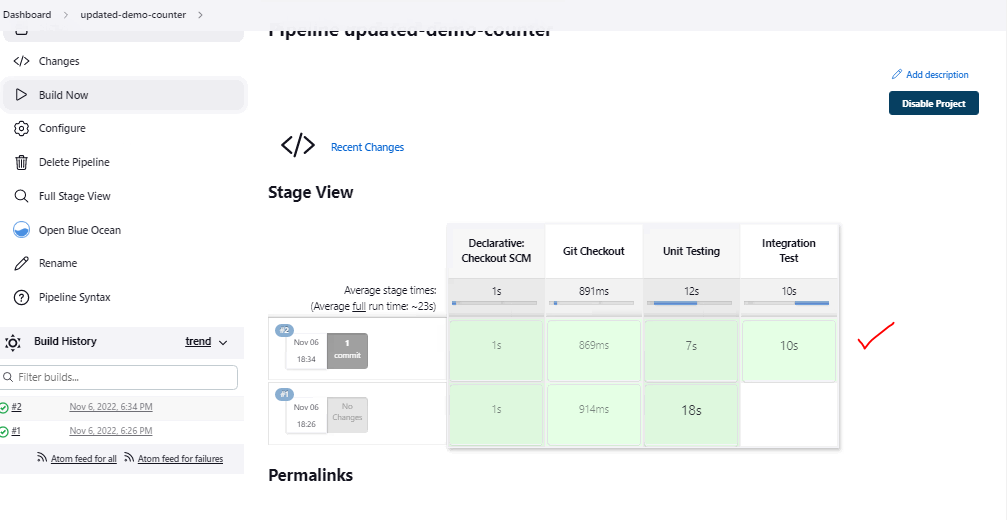
**Overall Updated Code**

|  |
| --- |
| **pipeline{**  **agent any**  **stages{**  **stage('Git Checkout'){**  **steps{**  **git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'**  **}**  **}**  **stage('Unit Testing'){**  **steps{**  **sh 'mvn test'**  **}**  **}**  **stage('Integration Test'){**  **steps{**  **sh 'mvn verify -DskipUnitTests '**  **}**  **}**  **}**  **}** |

**Push the Updated Code to Repository**

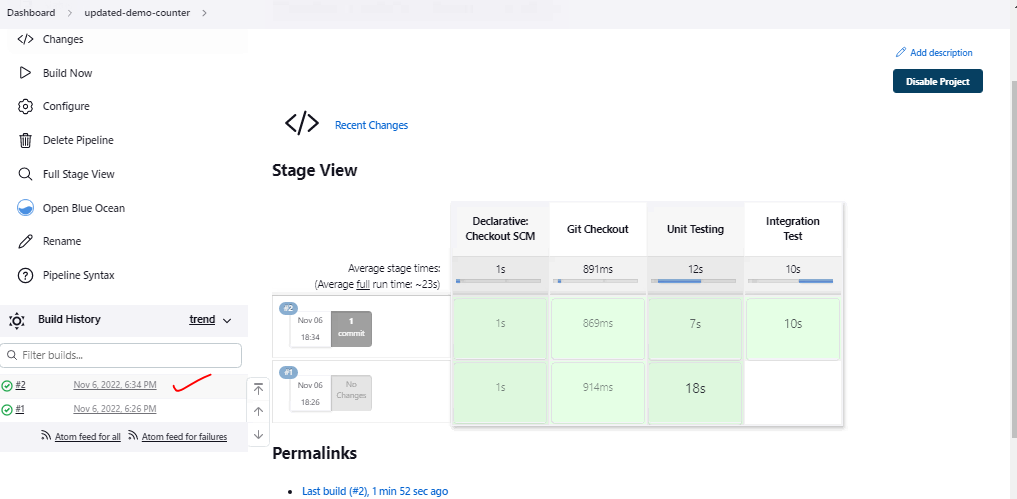


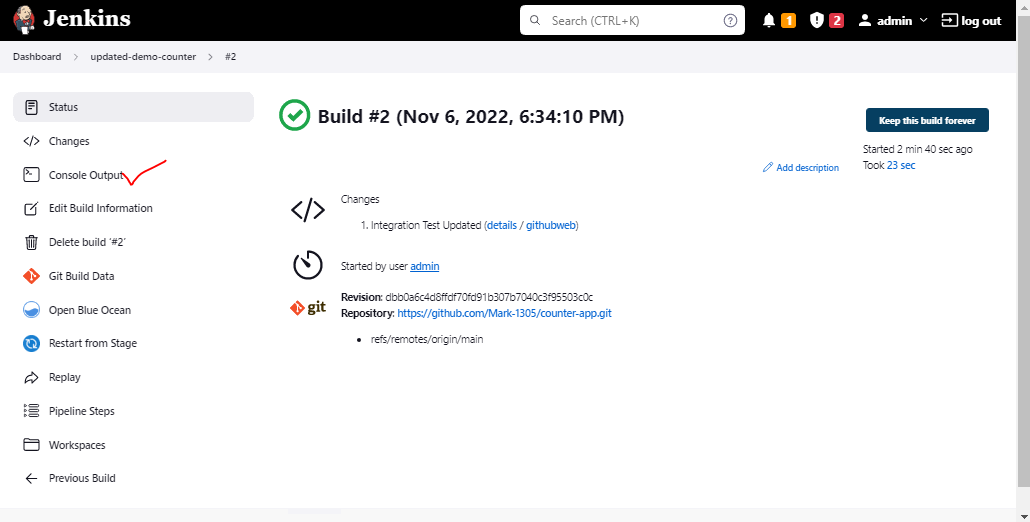
**Run the Pipeline**



**Integration Test Succeeded**

**Verify by Clicking on the Build Item**





**Sample Output**



**Now, we will move forward and build the Code**

**Stage: Build**

**Add the below code**

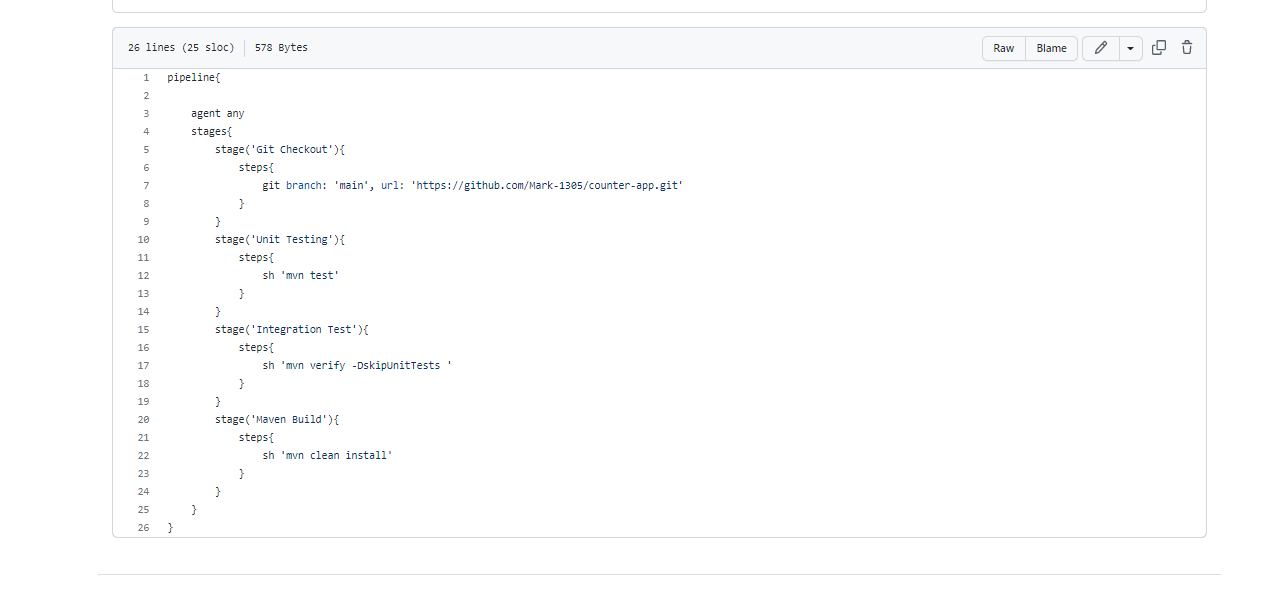
**stage('Maven Build'){**

**steps{**

**sh 'mvn clean install'**

**}**

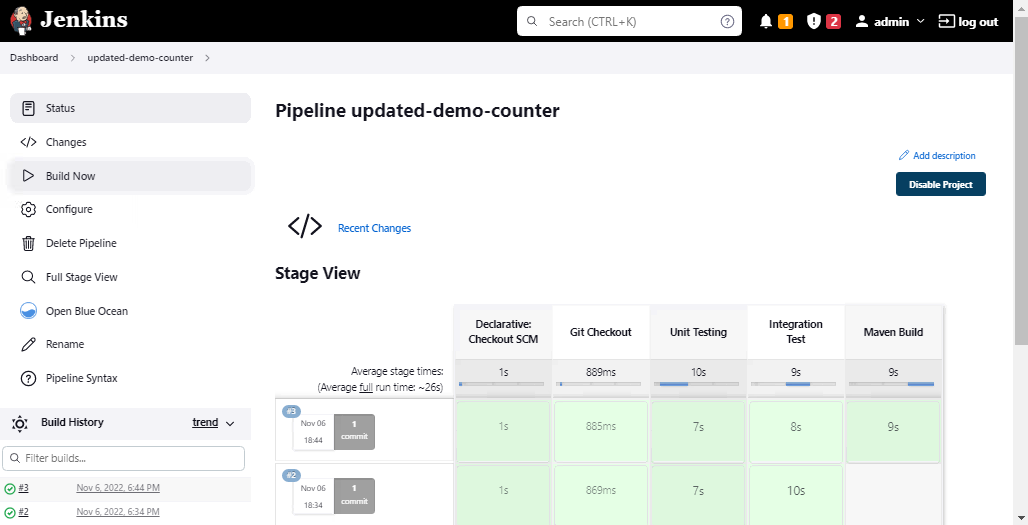
**}**



**The overall updated code**

|  |
| --- |
| **pipeline{**  **agent any**  **stages{**  **stage('Git Checkout'){**  **steps{**  **git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'**  **}**  **}**  **stage('Unit Testing'){**  **steps{**  **sh 'mvn test'**  **}**  **}**  **stage('Integration Test'){**  **steps{**  **sh 'mvn verify -DskipUnitTests '**  **}**  **}**  **stage('Maven Build'){**  **steps{**  **sh 'mvn clean install'**  **}**  **}**  **}**  **}** |

The Stage Successfully Run and the Code Build completed





**Stage: Static Code Analysis**

**Install SonarQube and Configure it with Jenkins**

**Configure SonarQube**

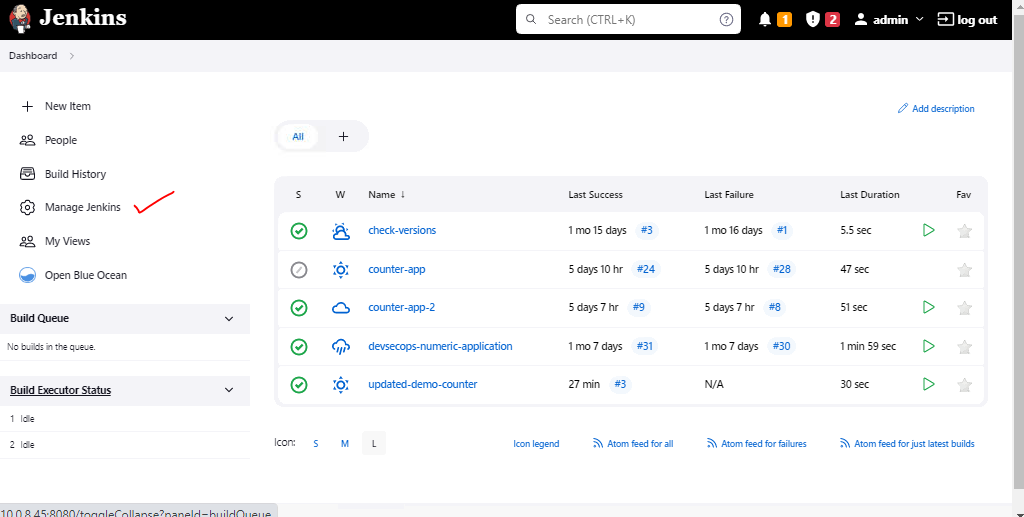
SonarQube:

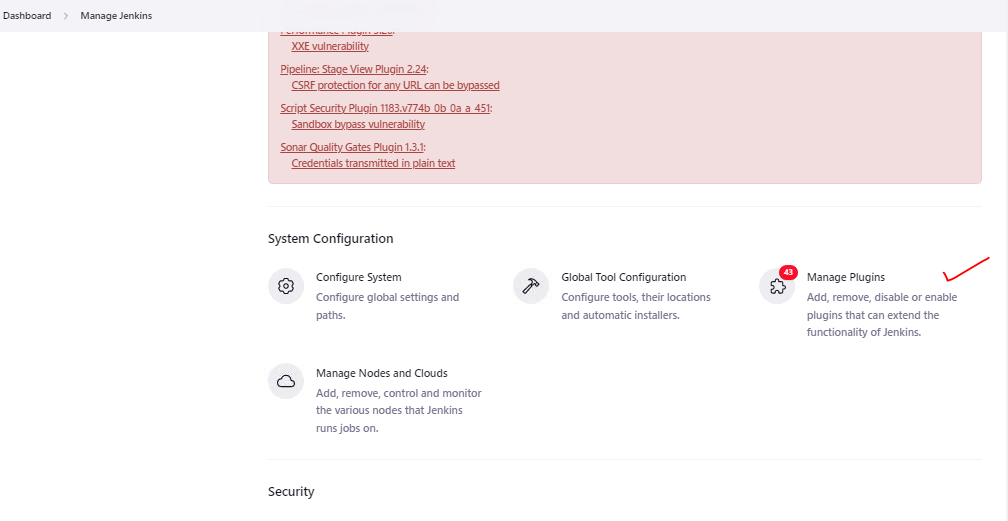
user: admin

password: admin123

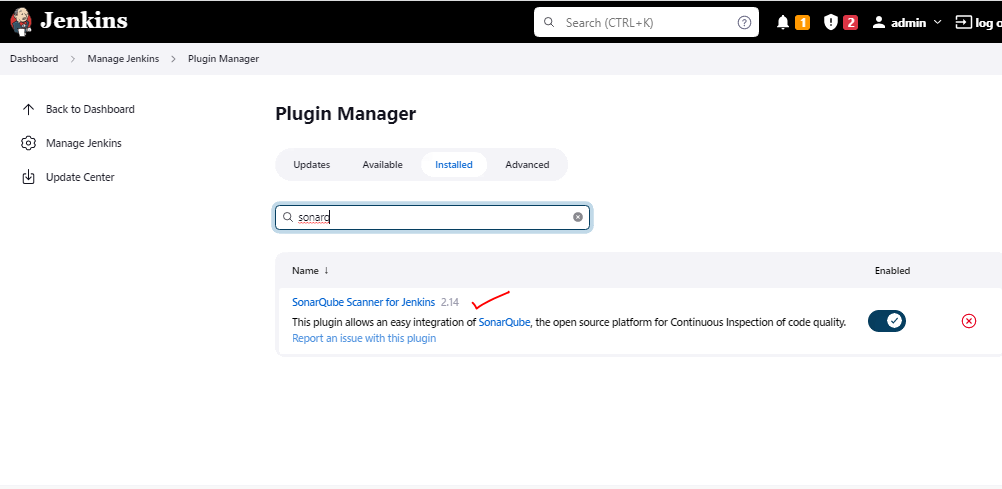
<http://10.0.8.45:9000/projects>

Add Sonarqube Plugin

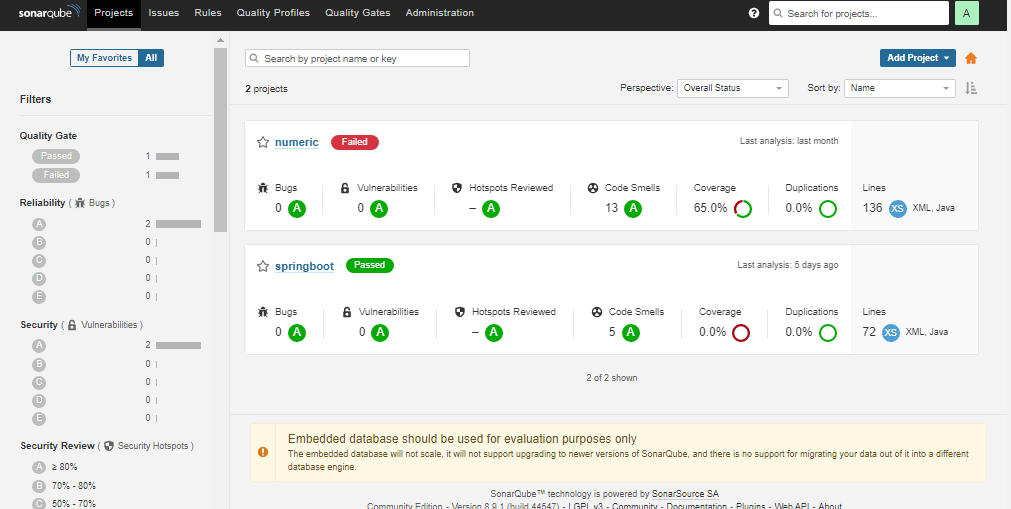




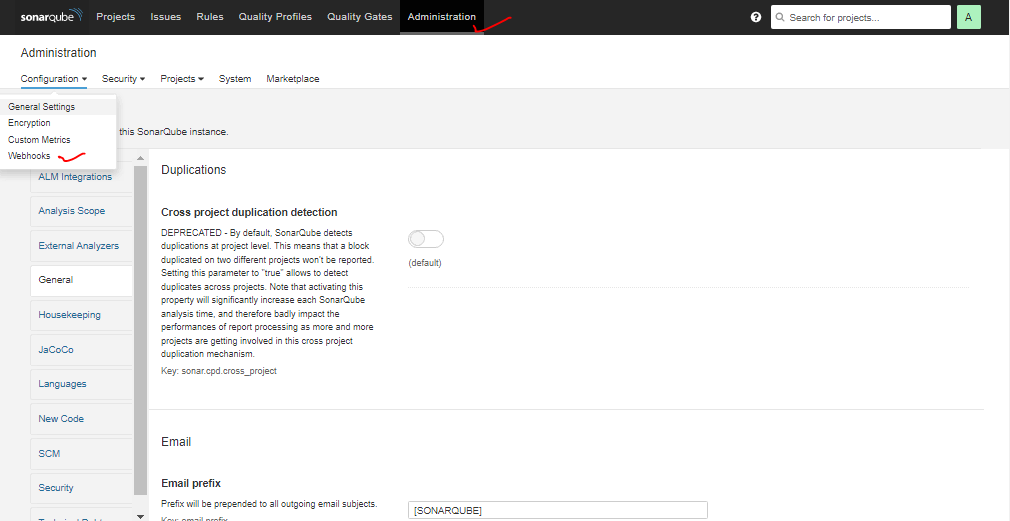
It’s already Installed

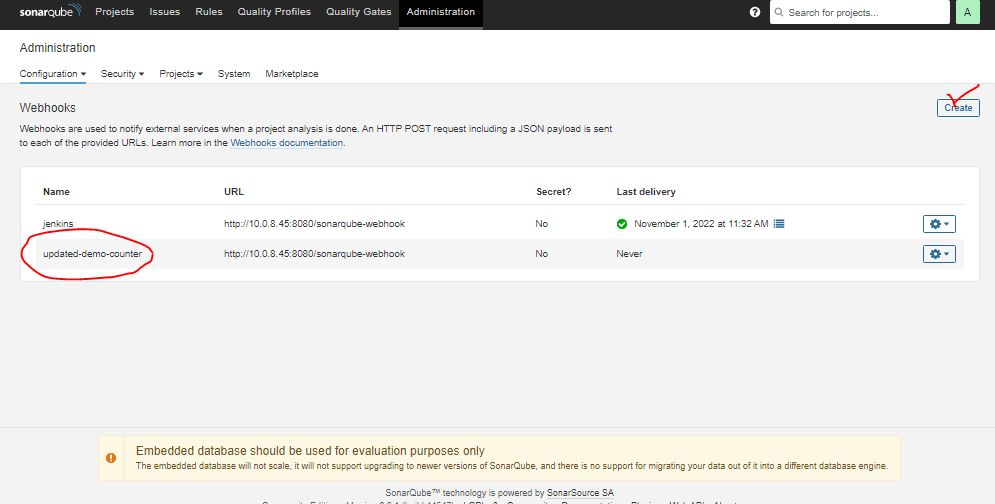


**Login to SonarQube Server**

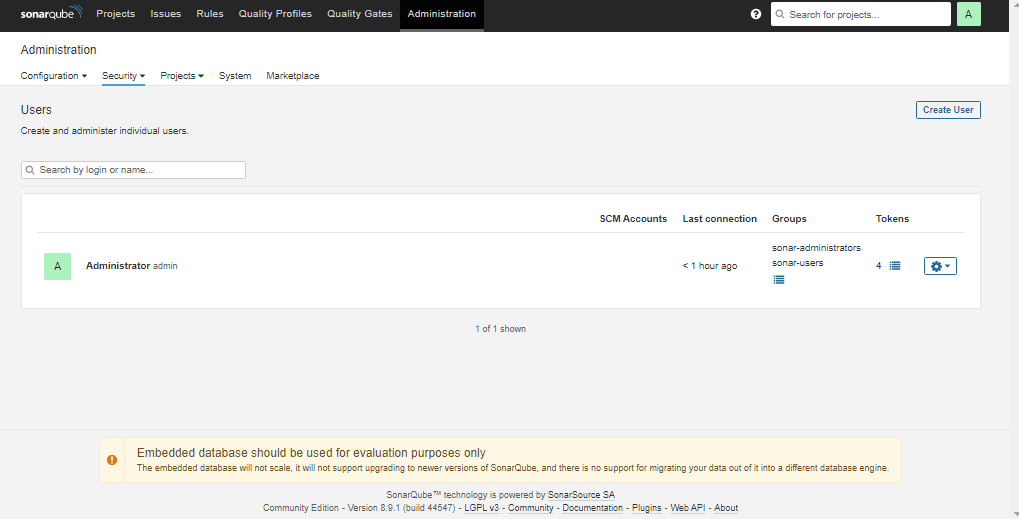


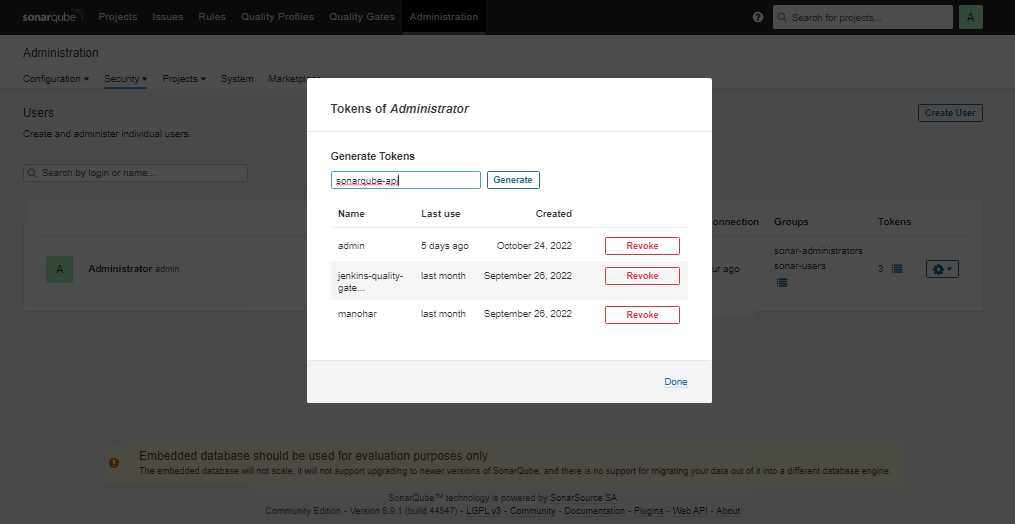
Go to Administartion - Webhook





Generate Token



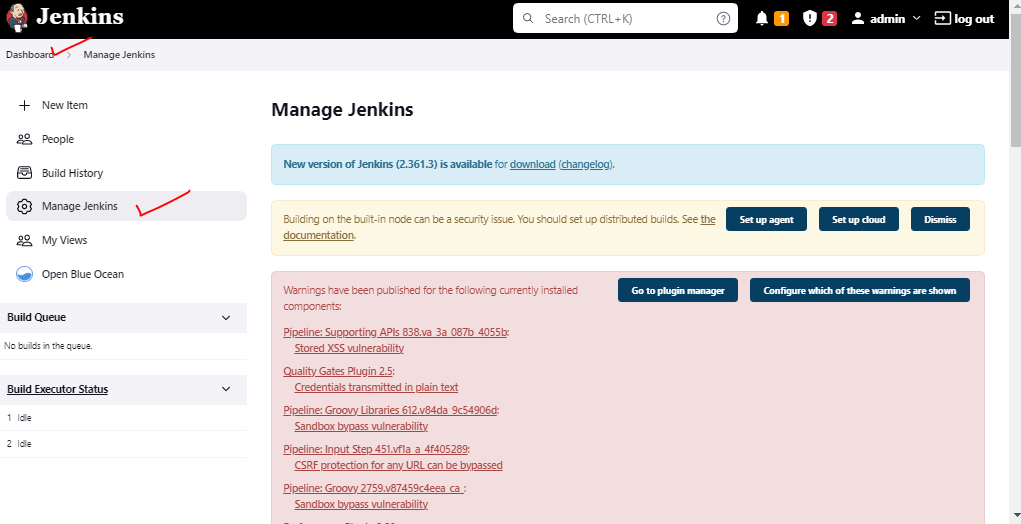


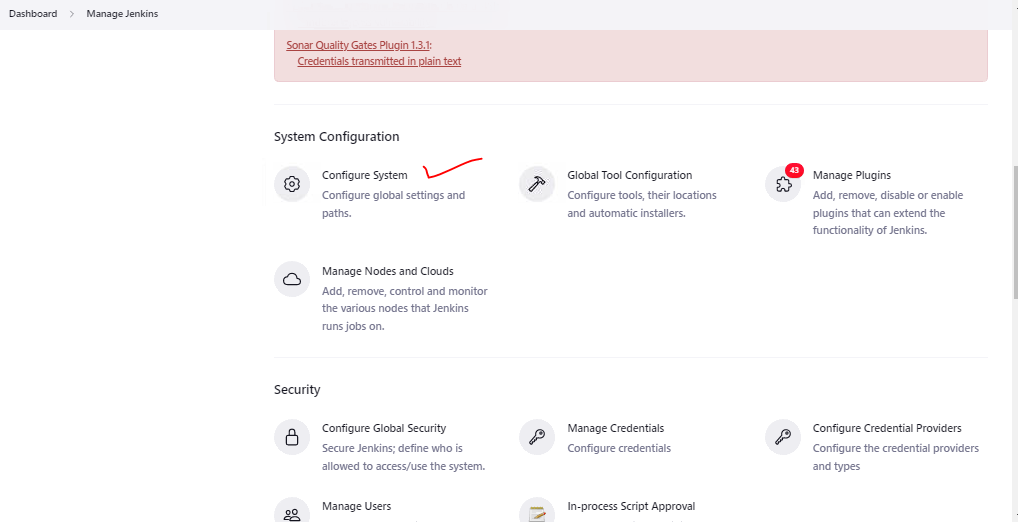
Token Id

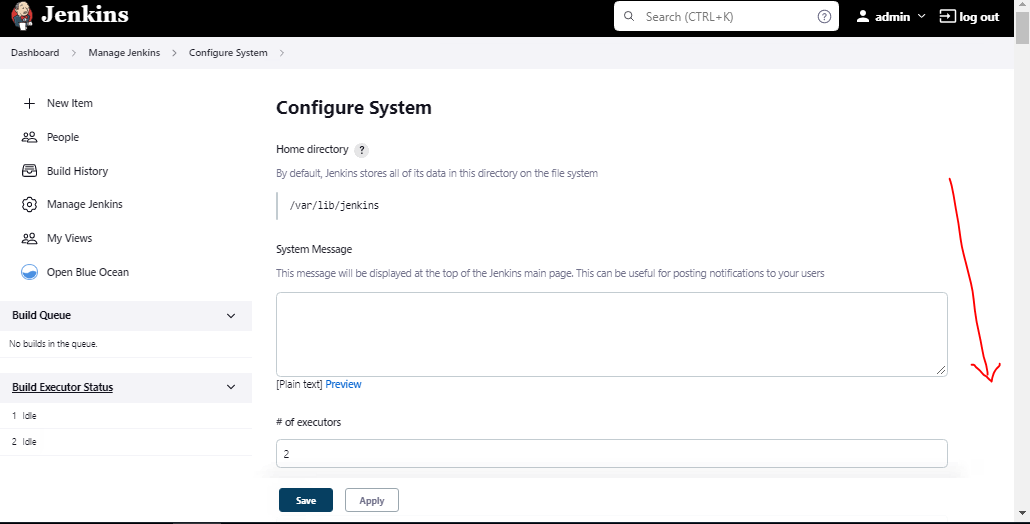
f1c1ca77946573f477f08d56cf8d6efd640c41ff

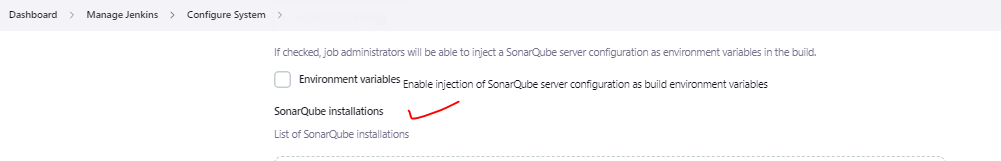
Add the token in Jenkins and Configure Sonarqube to Talk to Jenkins

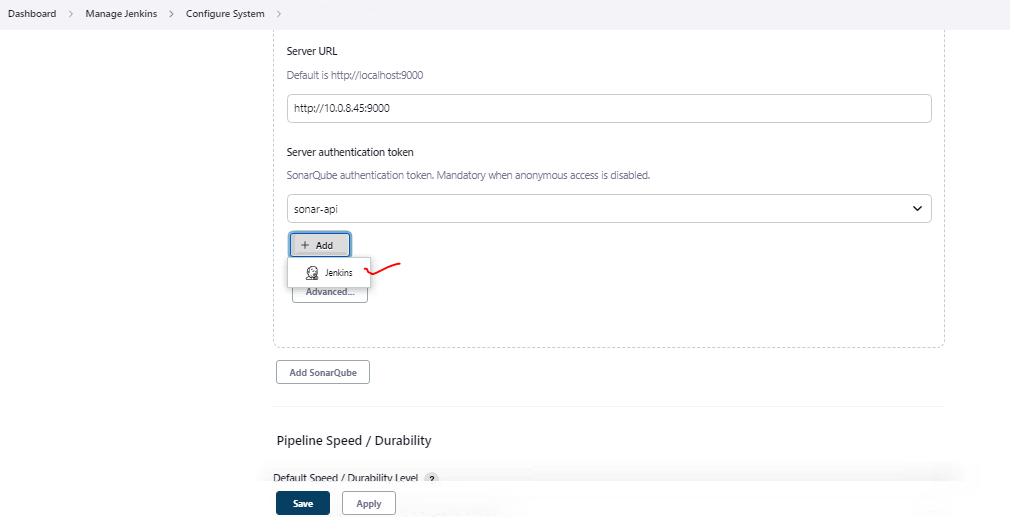
**Sonarqube Configuration in Jenkins**

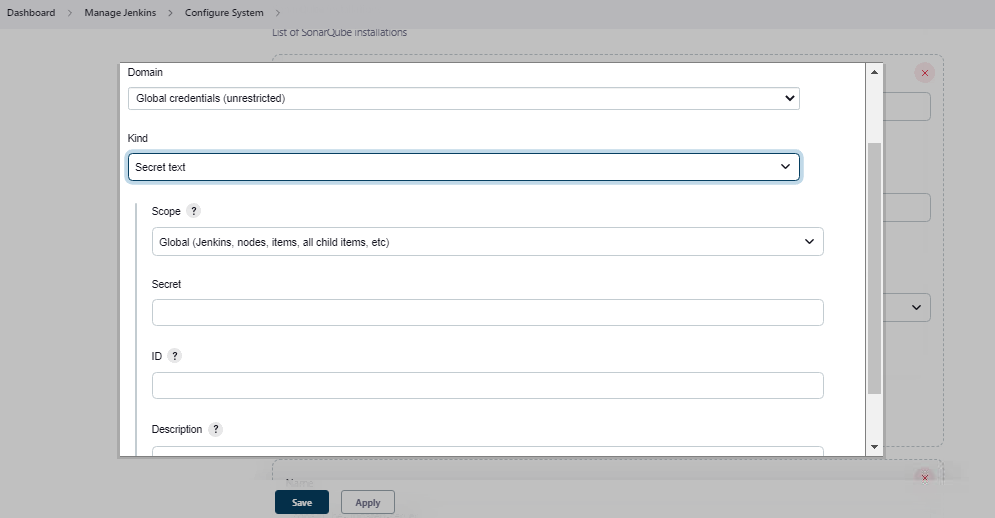


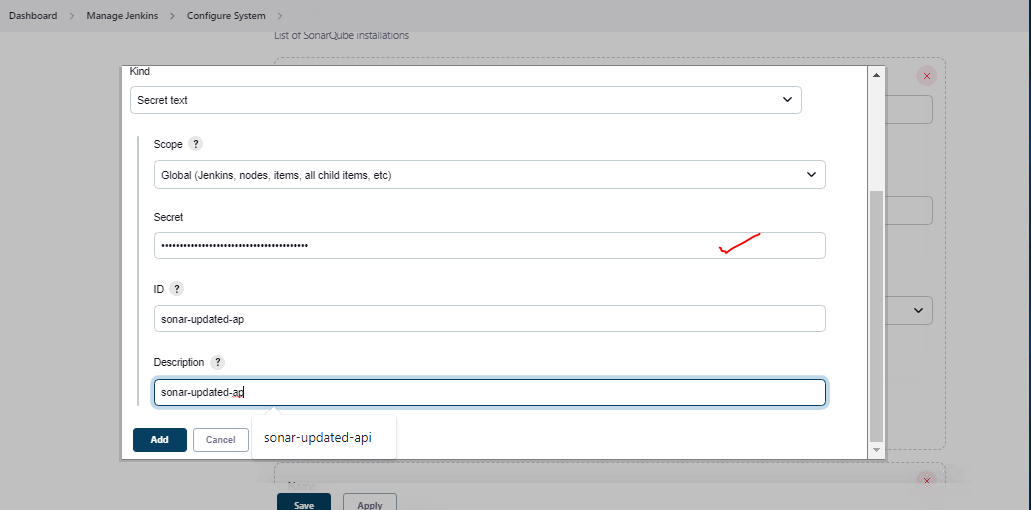




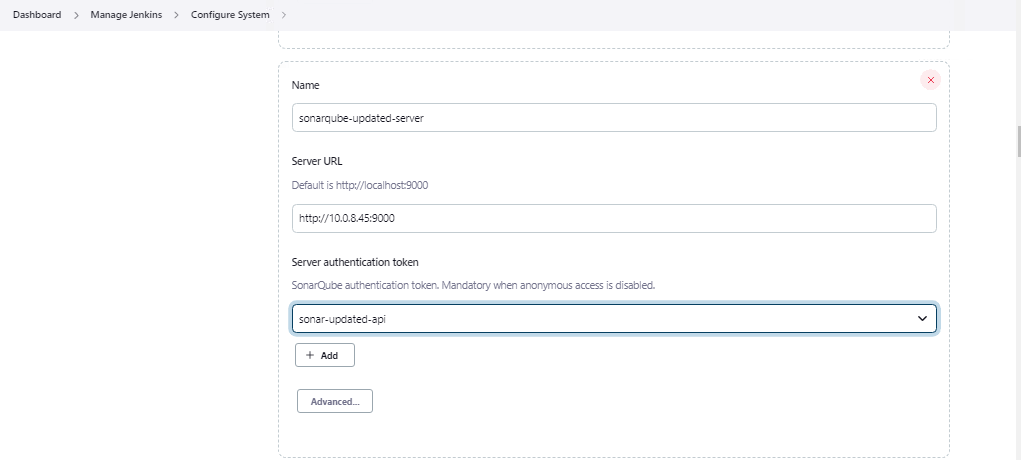








Add the Sonarqube Endpoint with the newly created Authentication to access the Sonarqube API

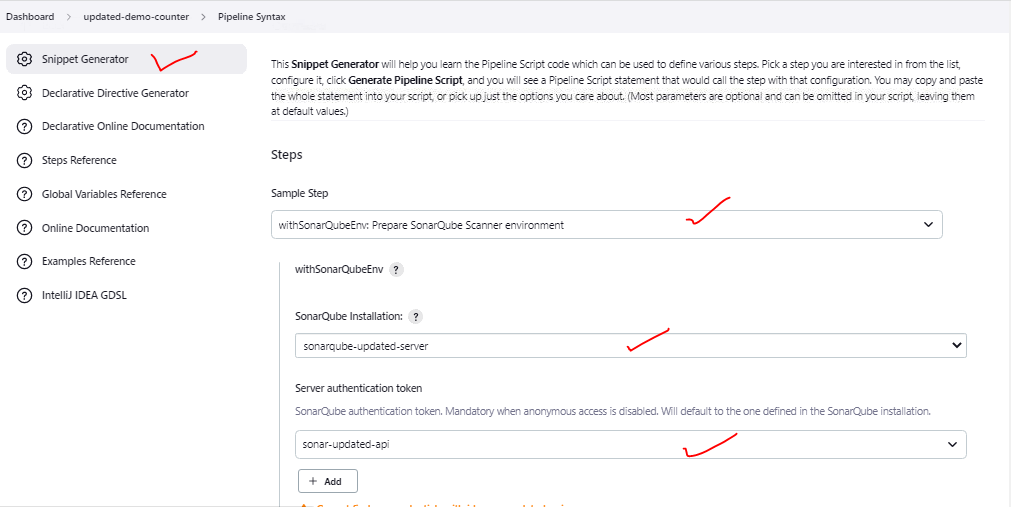


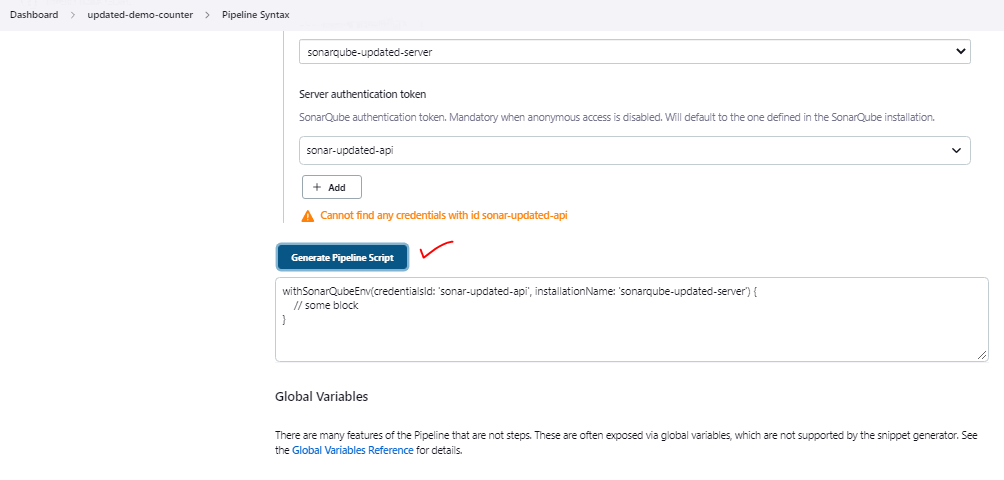
**Authetication Name: sonar-updated-api**

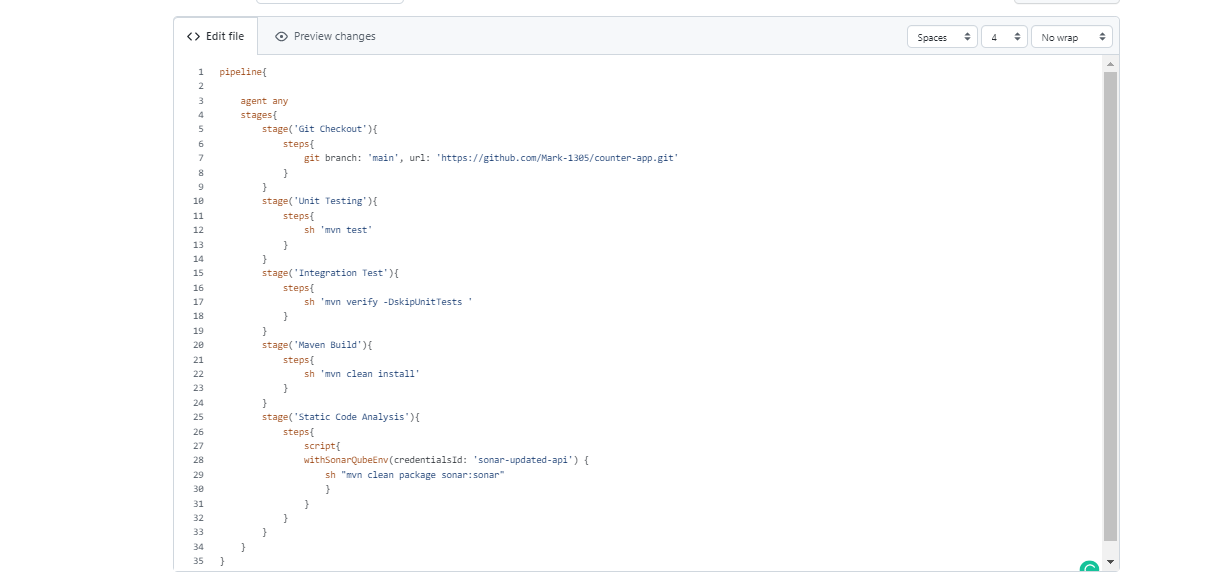
**Generate the pipeline script for the SonarQube – Static Code Anlaysis**

**Generated Code**

|  |
| --- |
| withSonarQubeEnv(credentialsId: 'sonar-updated-api', installationName: 'sonarqube-updated-server') |

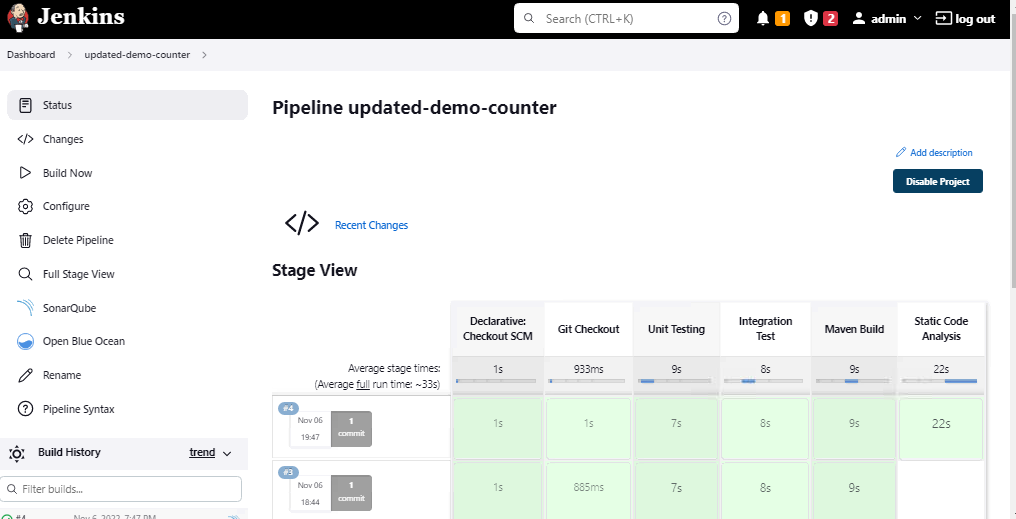


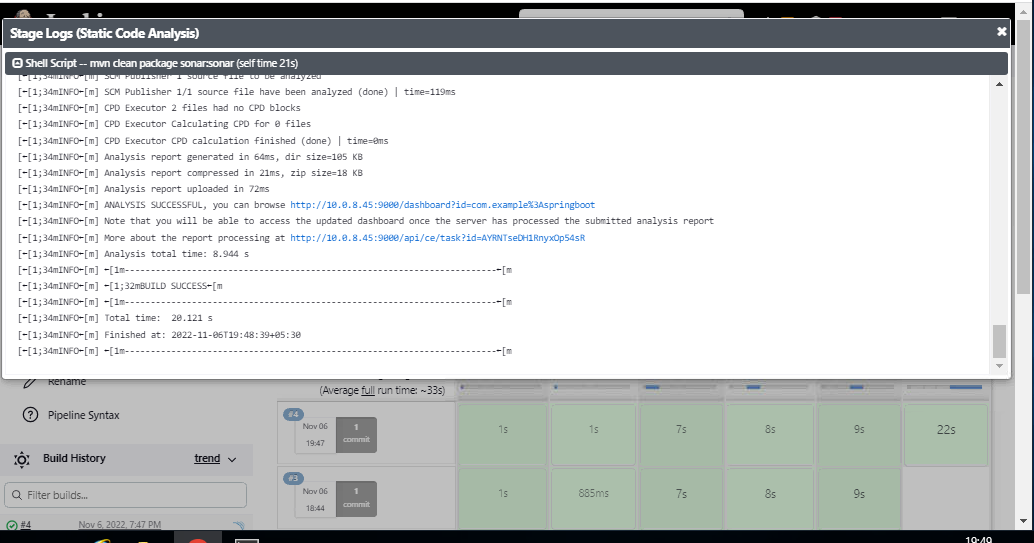


**Overall Updated Code**

|  |
| --- |
| **pipeline{**  **agent any**  **stages{**  **stage('Git Checkout'){**  **steps{**  **git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'**  **}**  **}**  **stage('Unit Testing'){**  **steps{**  **sh 'mvn test'**  **}**  **}**  **stage('Integration Test'){**  **steps{**  **sh 'mvn verify -DskipUnitTests '**  **}**  **}**  **stage('Maven Build'){**  **steps{**  **sh 'mvn clean install'**  **}**  **}**  **stage('Static Code Analysis'){**  **steps{**  **script{**  **withSonarQubeEnv(credentialsId: 'sonar-updated-api') {**  **sh "mvn clean package sonar:sonar"**  **}**  **}**  **}**  **}**  **}**  **}** |

**Update and Run the Pipeline**



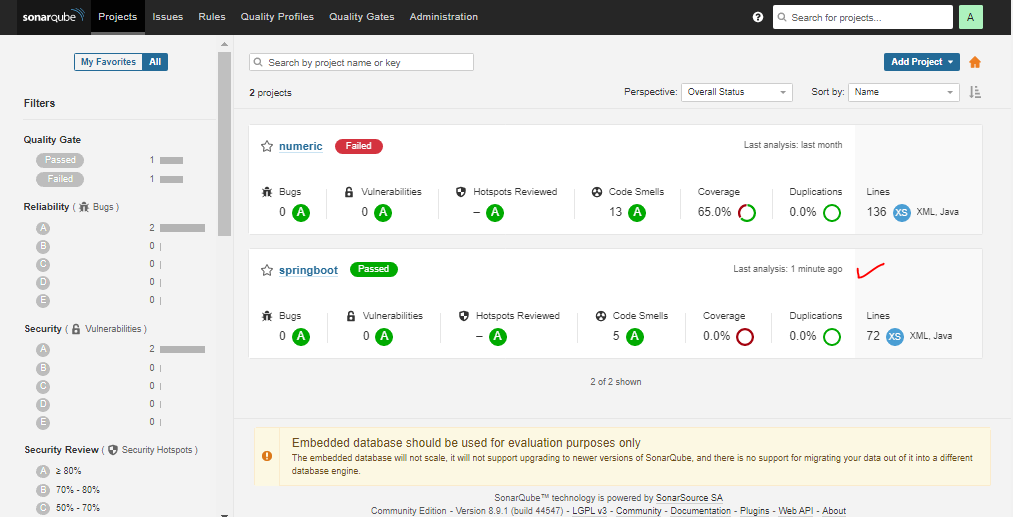


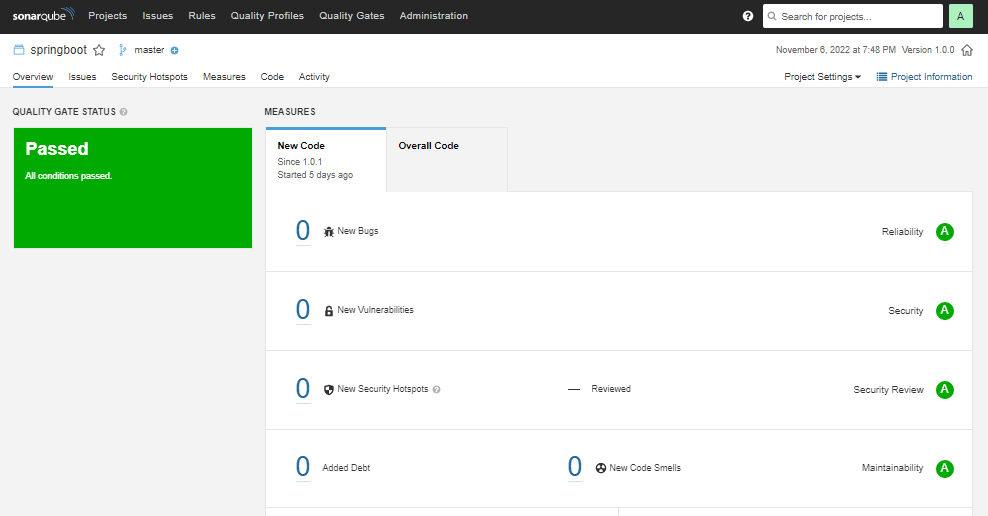
**Check the Results in SonarQube Dashboard**

The the stage in Jenkins will connect the Sonarqube API using the token generated in Sonarqube -Administration-Configuration-Security Path.

As this token is been passed in Jenkins in Mange-jenkins-Configure-System-Sonarqube-New-Secret-text was created and using this authentication method.

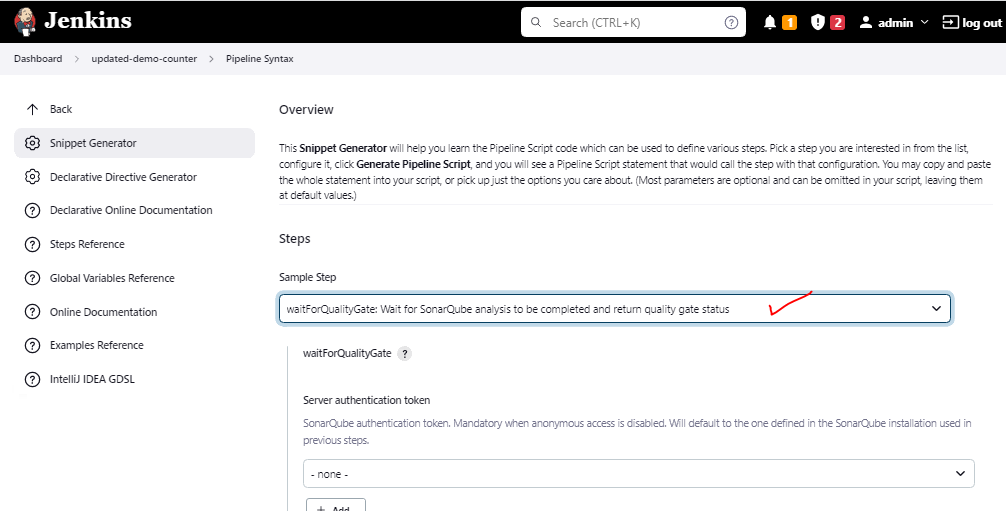
Jenkins was able to talk to Sonarqube and do the Static-Code- Analysis of the code in the pipeline

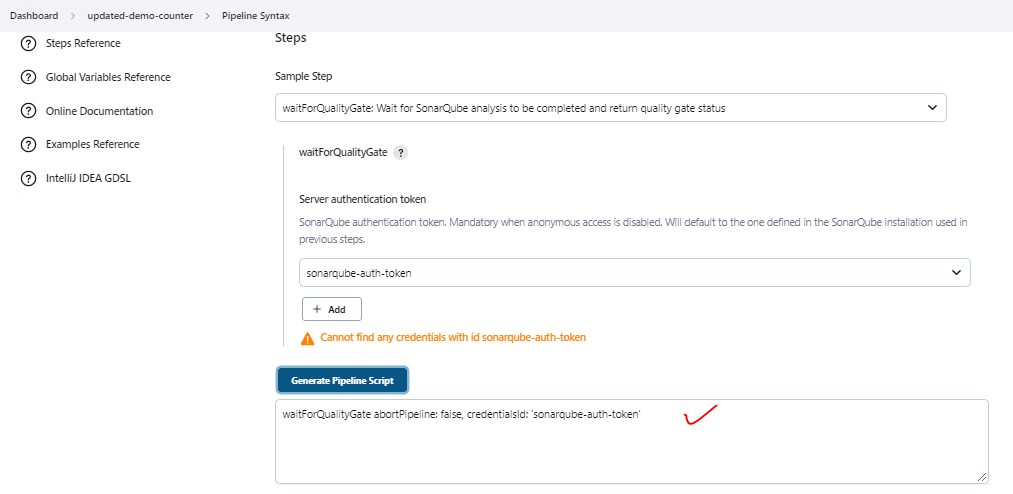




**Stage: Quality-Gate-Status**

**Generate the Script using Pipeline Syntax**





**Generated Script**

**waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token'**

**Add below Lines**

stage('Quality Gate Status'){

steps{

script{

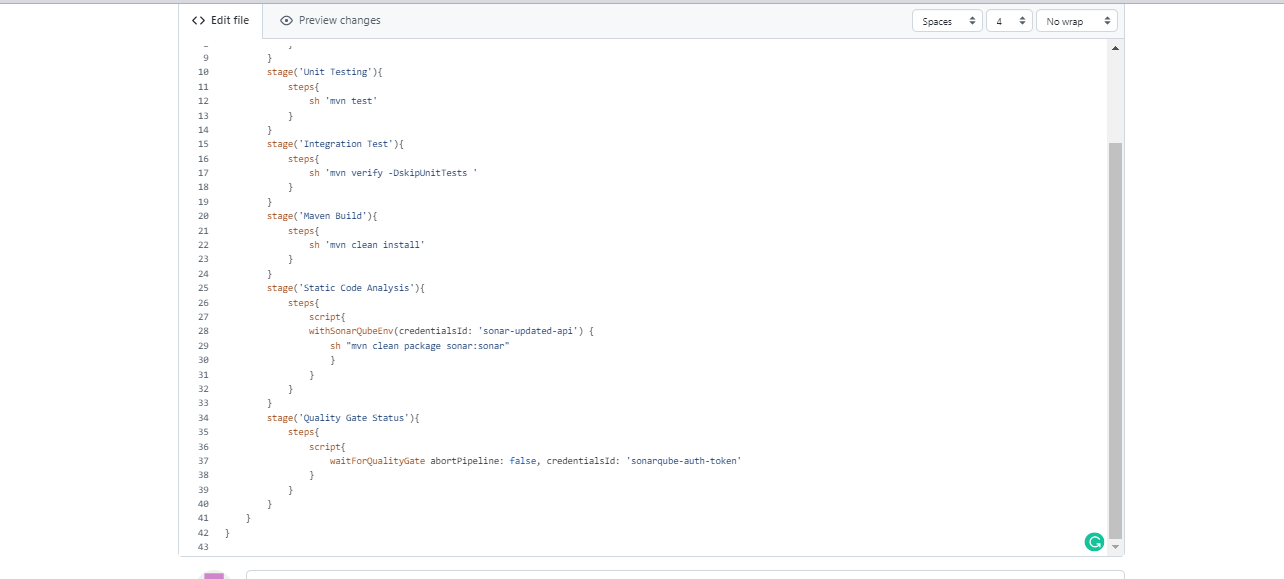
waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token'

}

}

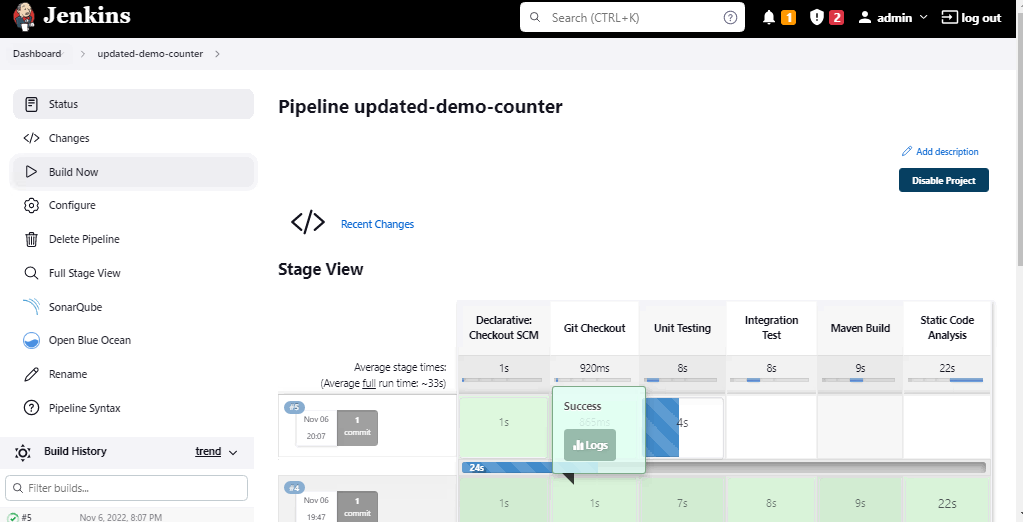
}

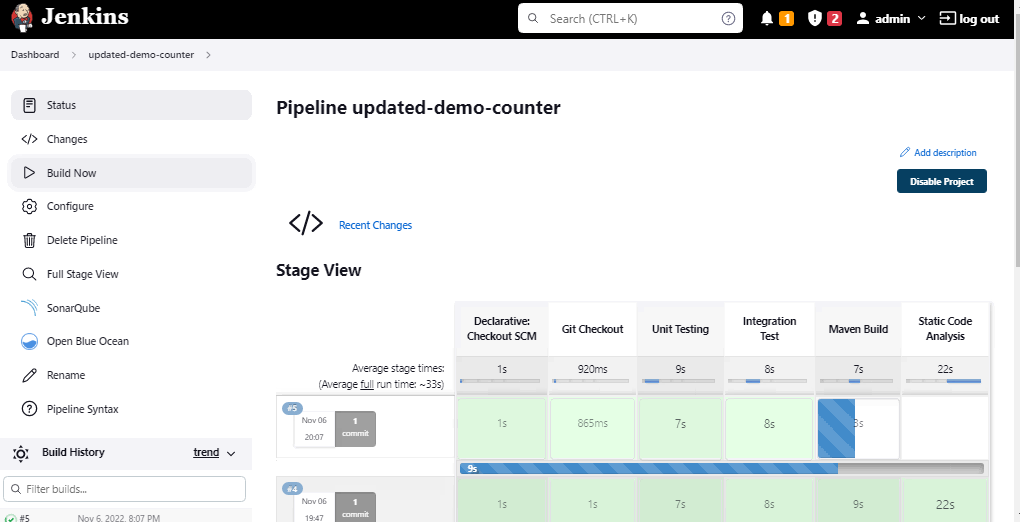
**Overall Code**

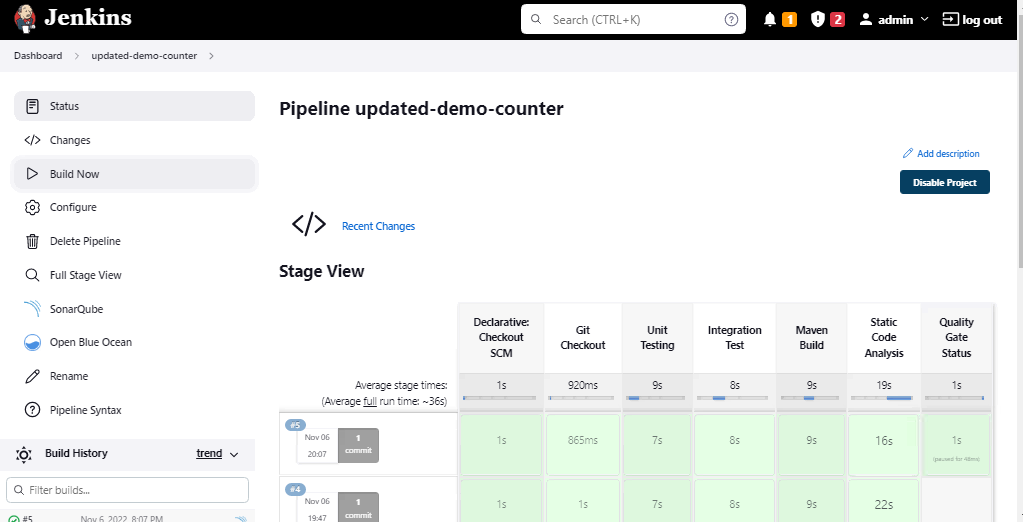


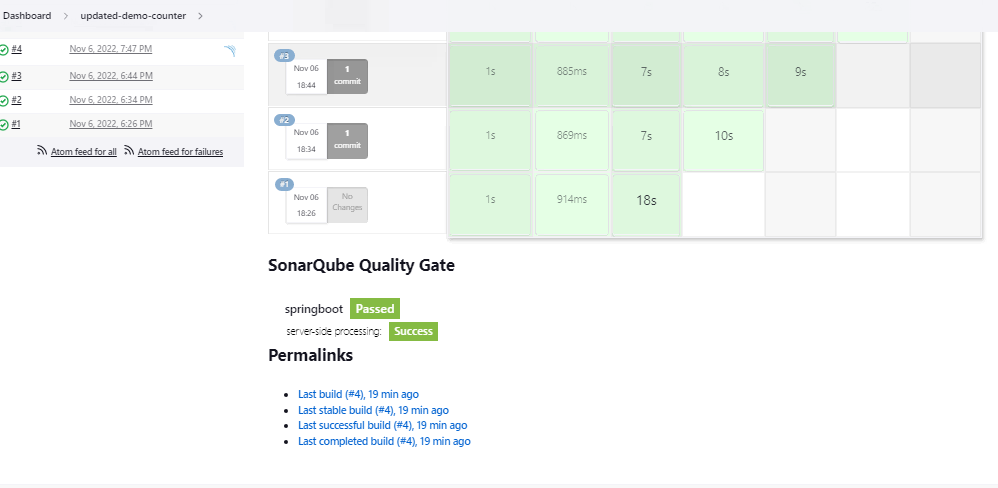
|  |
| --- |
| pipeline{  agent any  stages{  stage('Git Checkout'){  steps{  git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'  }  }  stage('Unit Testing'){  steps{  sh 'mvn test'  }  }  stage('Integration Test'){  steps{  sh 'mvn verify -DskipUnitTests '  }  }  stage('Maven Build'){  steps{  sh 'mvn clean install'  }  }  stage('Static Code Analysis'){  steps{  script{  withSonarQubeEnv(credentialsId: 'sonar-updated-api') {  sh "mvn clean package sonar:sonar"  }  }  }  }  stage('Quality Gate Status'){  steps{  script{  waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token'  }  }  }  }  } |

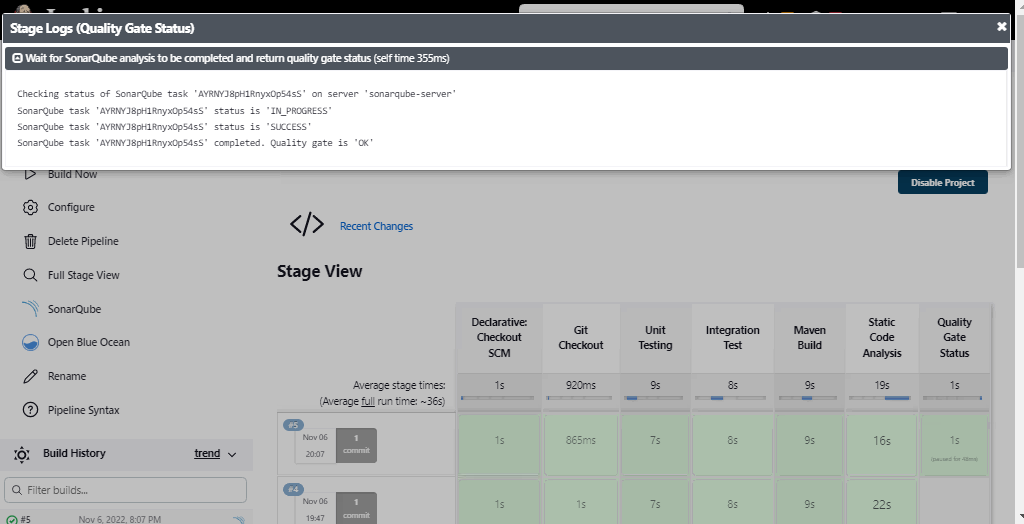
**Run the pipeline**

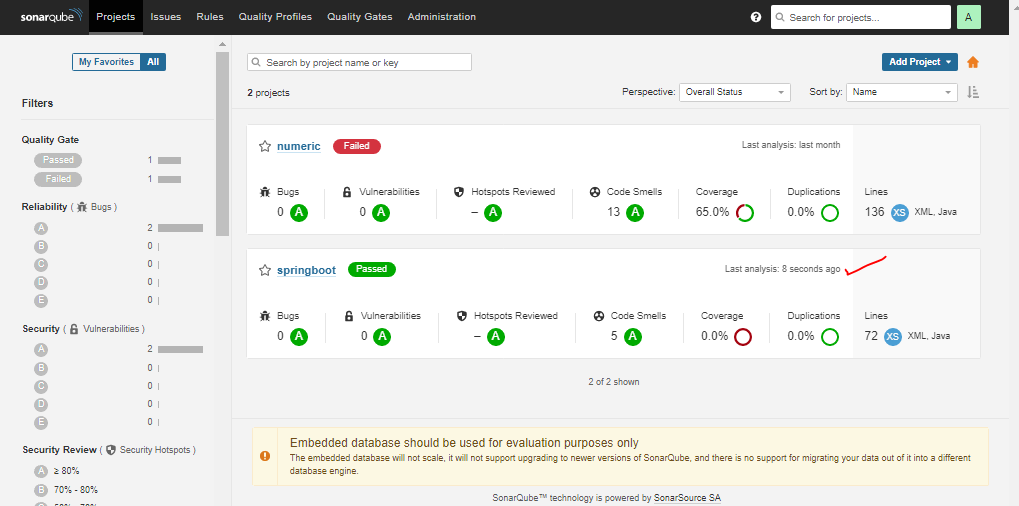












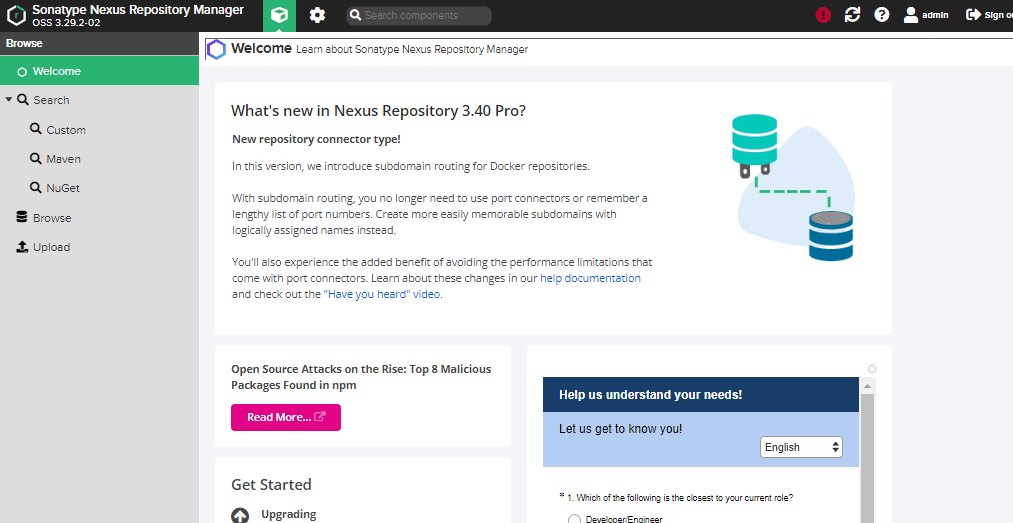
**The Stage ran successfully**

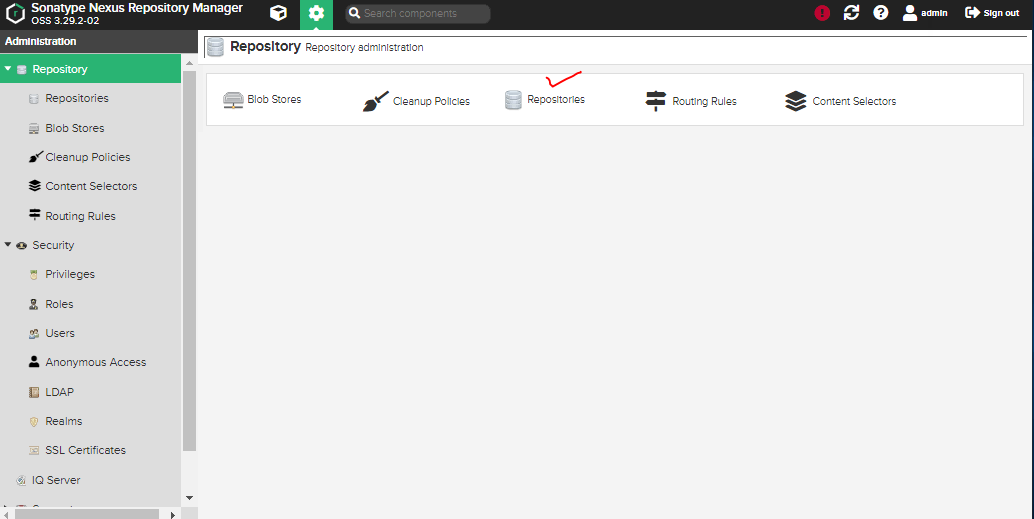
**NEXUS Repository \_ Push the Artifacts**

**Save the generated Artifacts into a Nexus Repository**

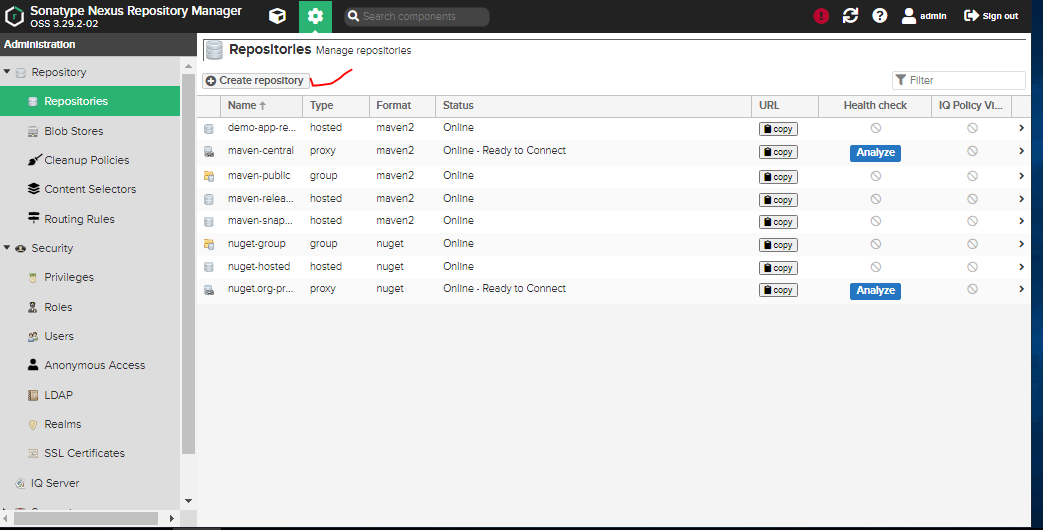
**Create a Nexus Server Separately**

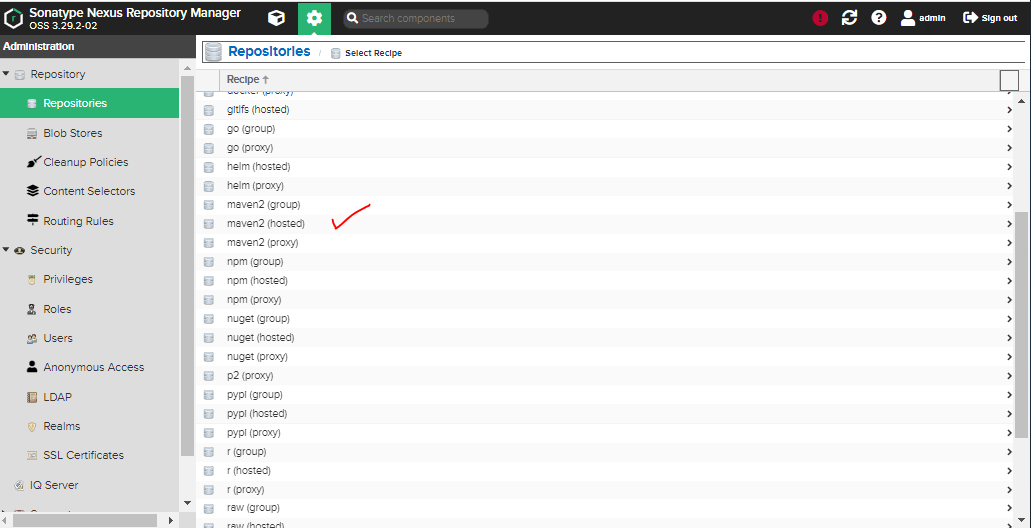
**Login to the server**

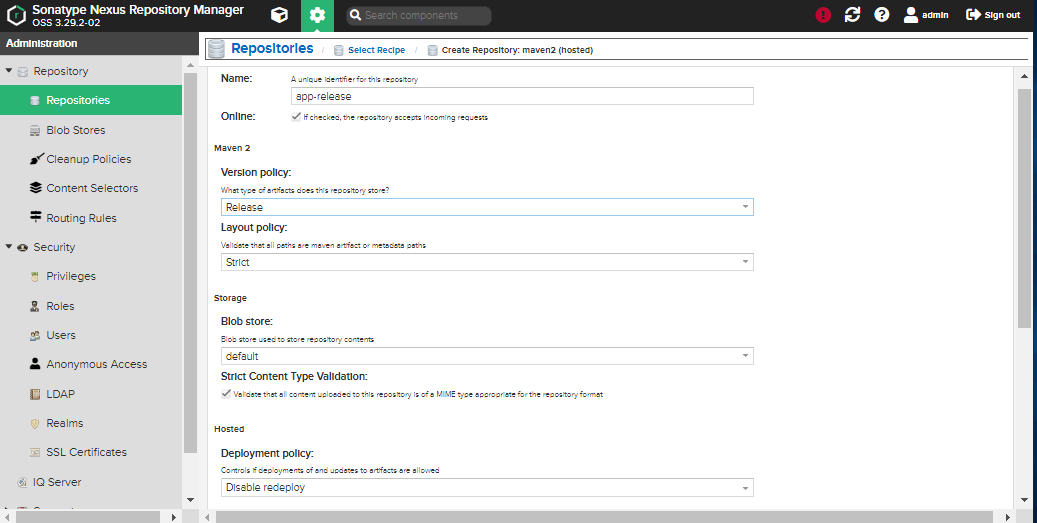


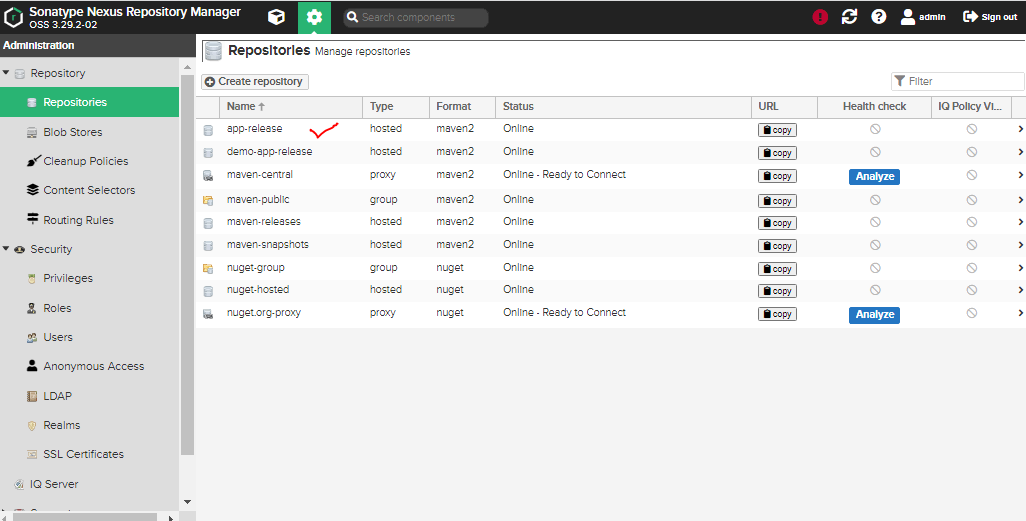


**Create a Repo – Custome Repository**

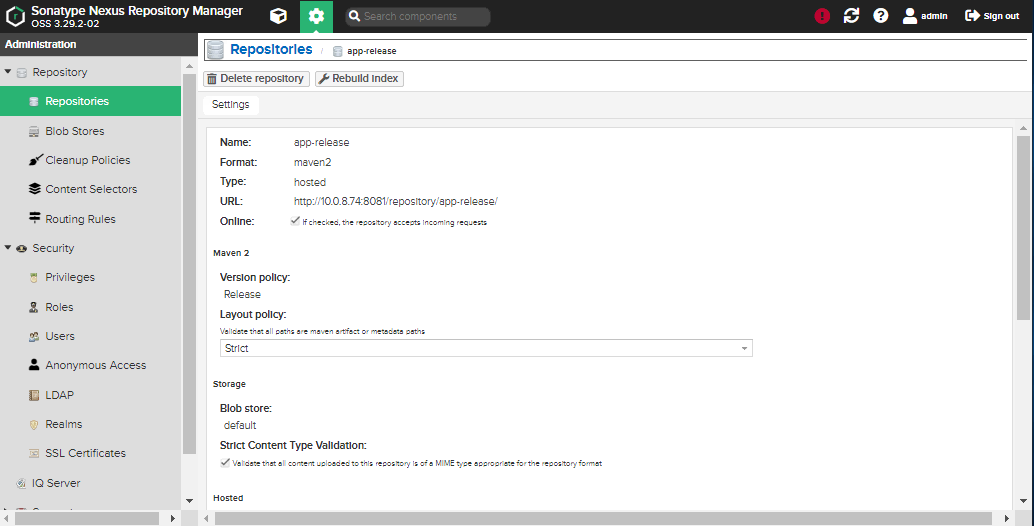






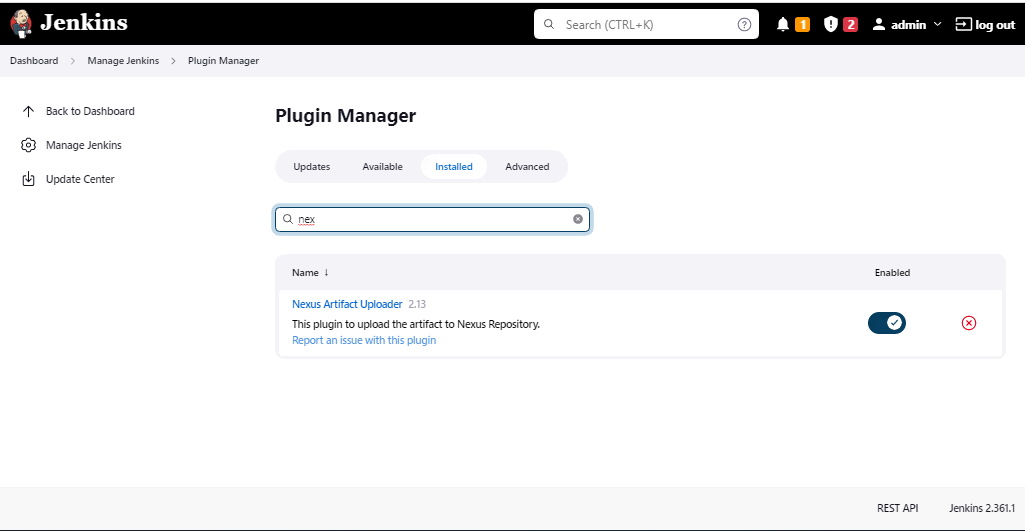


**Keep the details**

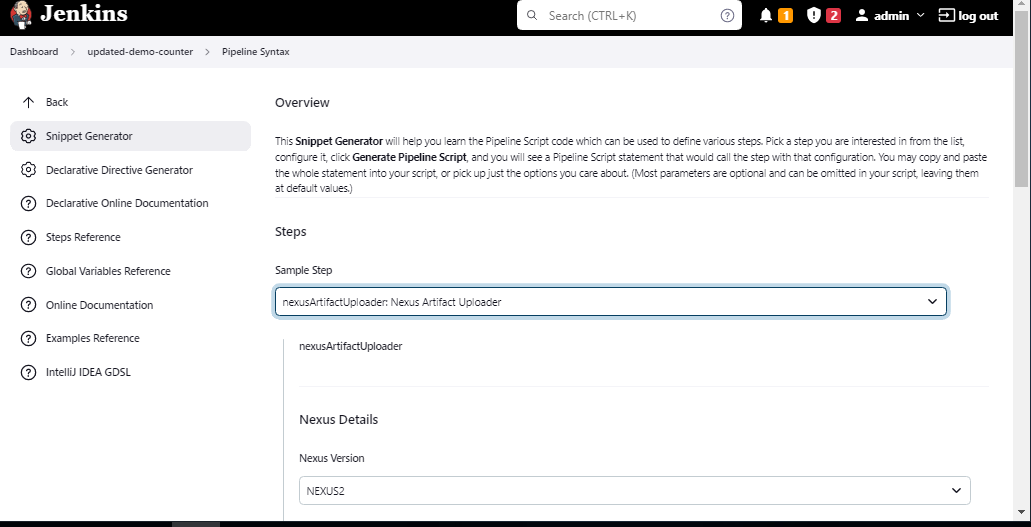


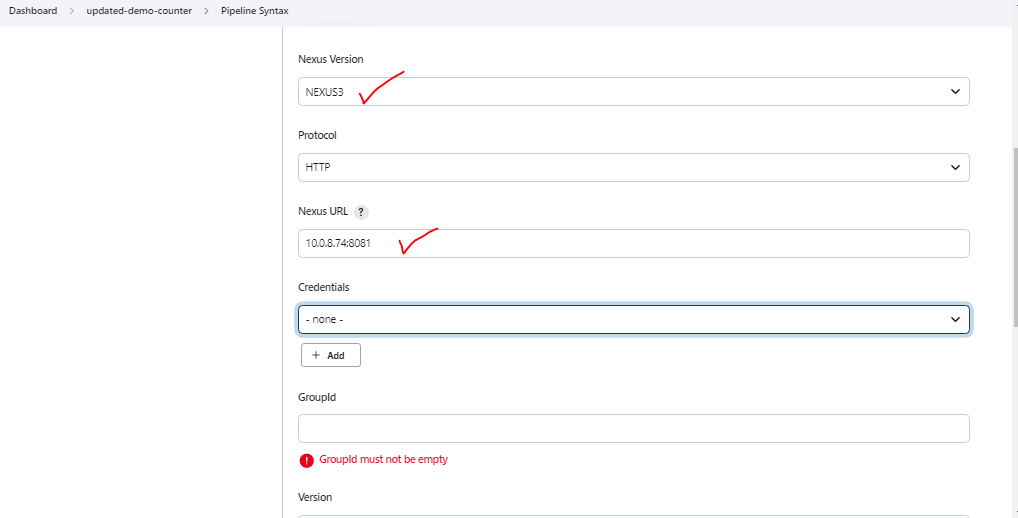
**Now, make a connection between Jenkins and Nexus so that the API call happens and the artifacts gets stored in Nexus during the build stage**

**Install a Plugin in Jenkins for Nexus**

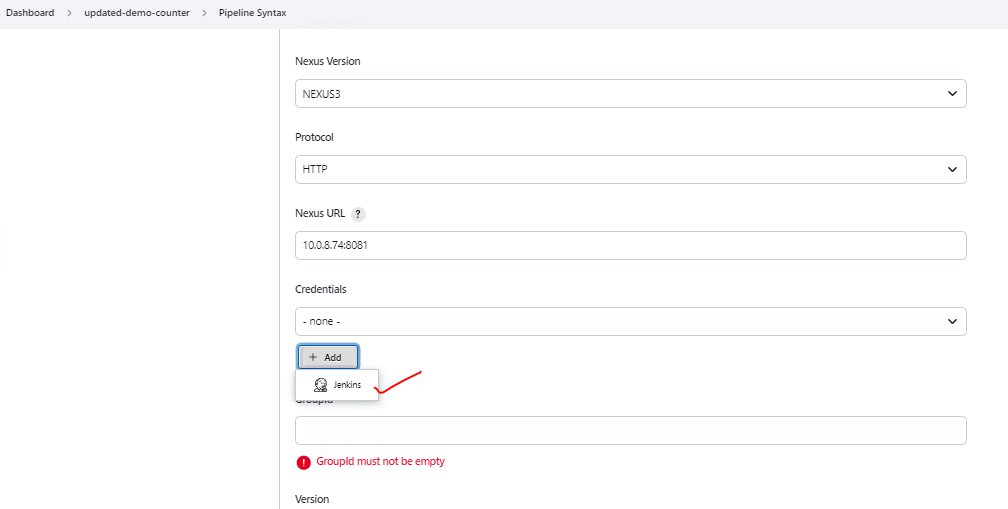


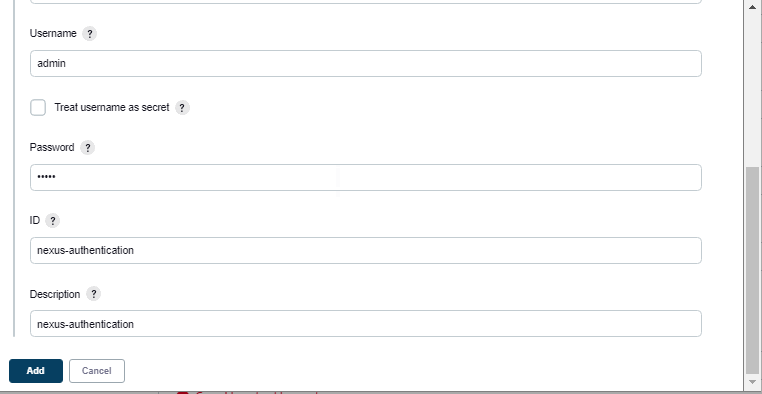
**Create a Pipeline script for Nexus to add in Jenkinsfile**



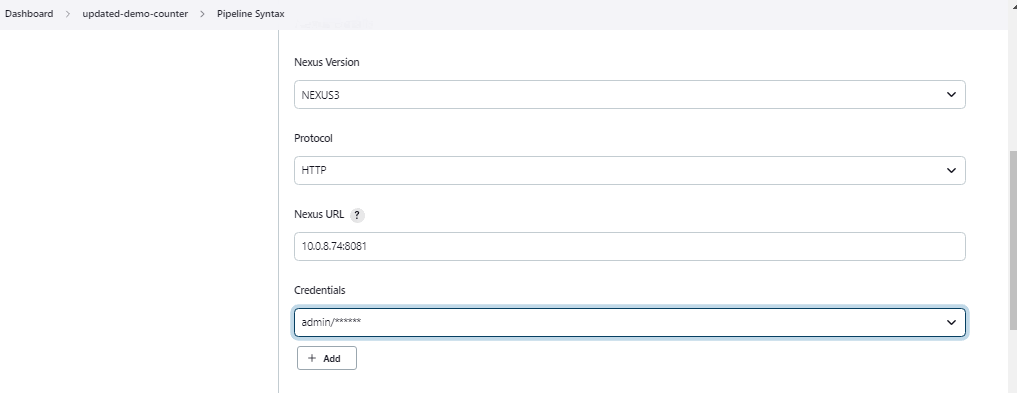


**Add Credentials**



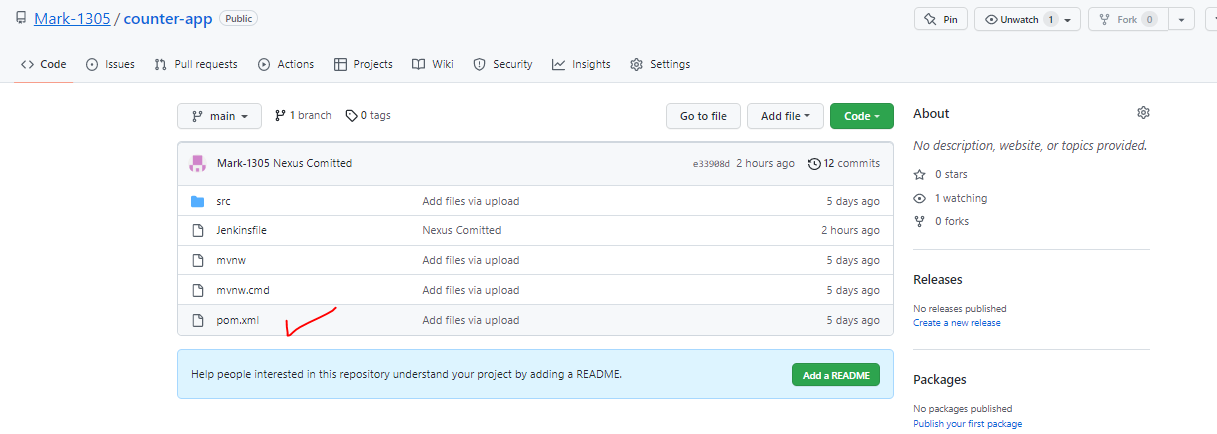


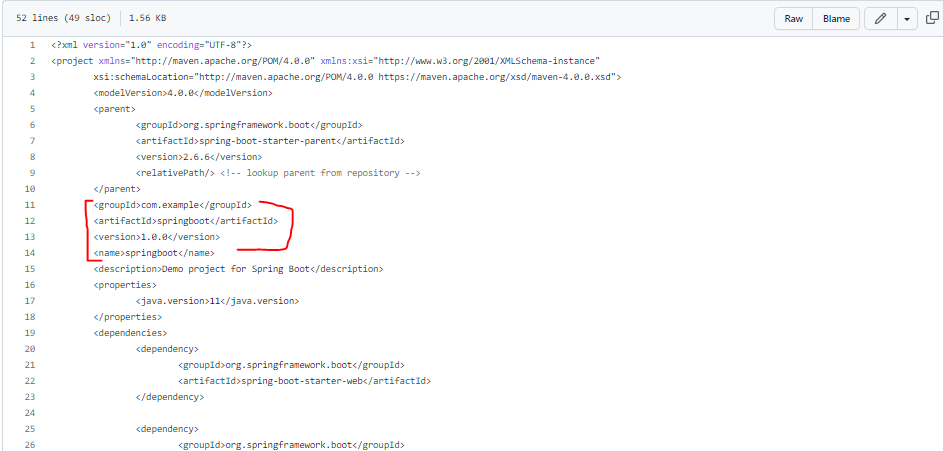
**The Created Credentials are Selected**

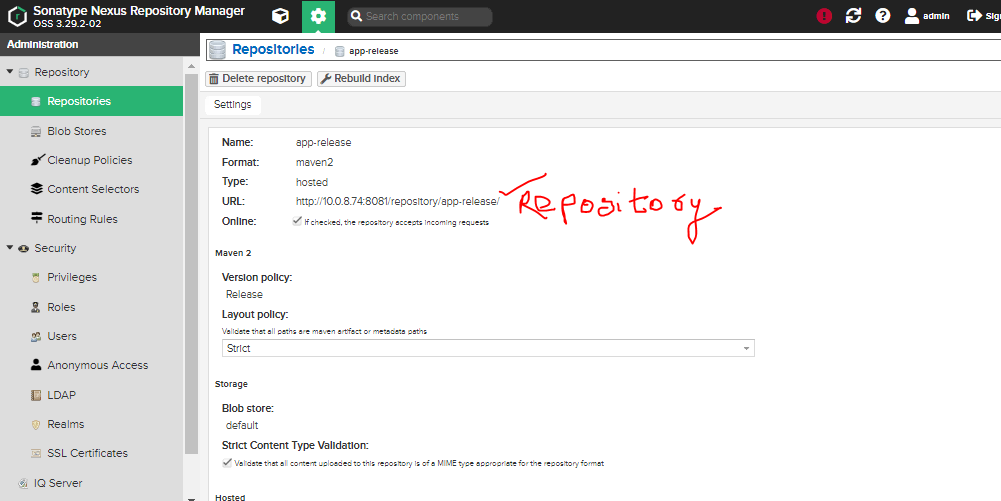


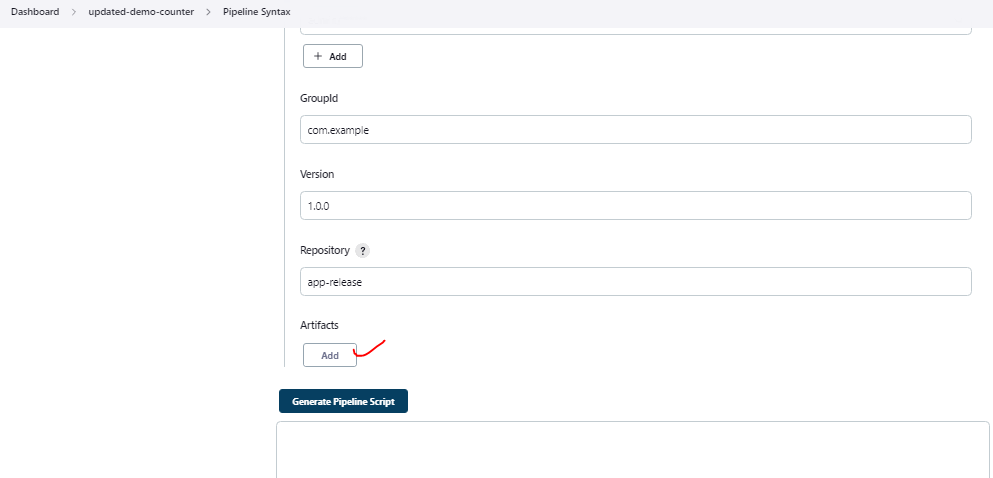
**Add Group ID by Using POM.Xml – Get into the code**

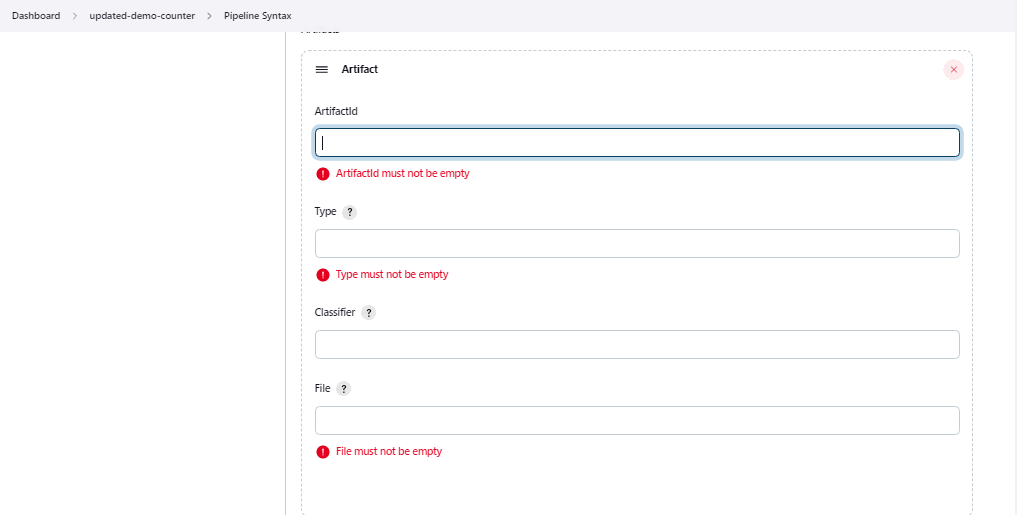








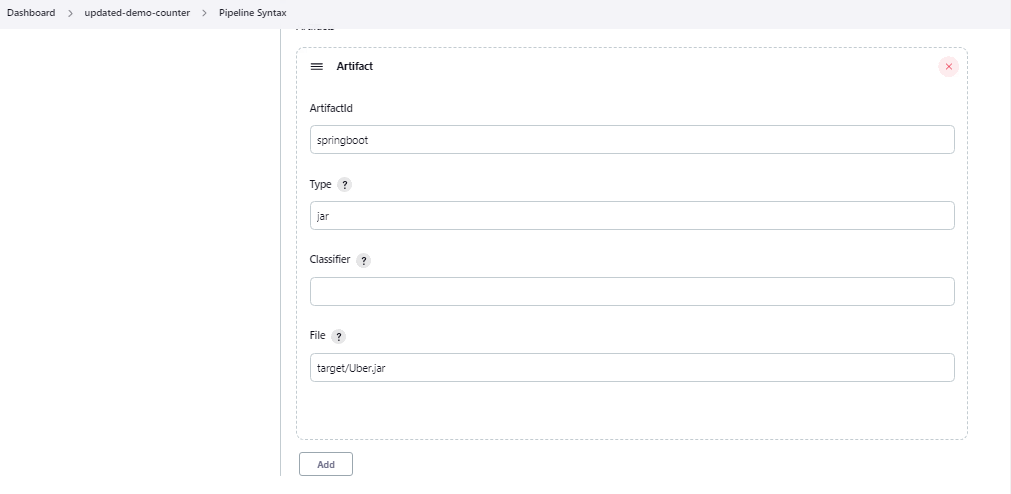




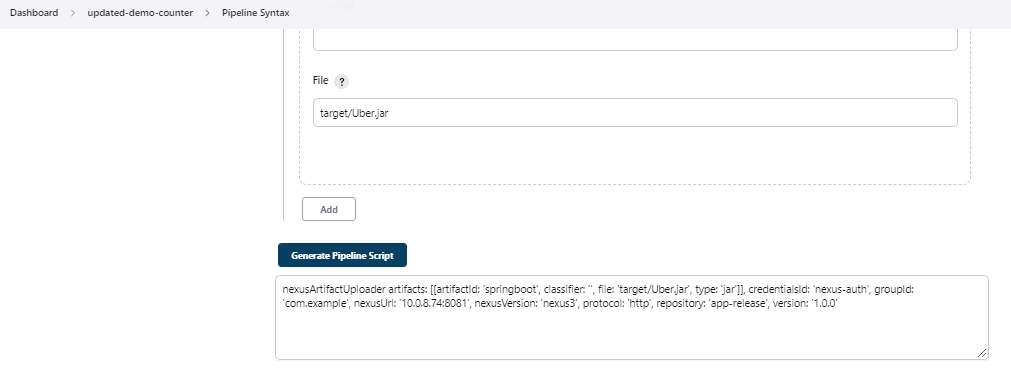


**Check the Path where Maven stored the jar files**





**Script Generated**



**Generated Script**

**nexusArtifactUploader artifacts: [[artifactId: 'springboot', classifier: '', file: 'target/Uber.jar', type: 'jar']], credentialsId: 'nexus-auth', groupId: 'com.example', nexusUrl: '10.0.8.74:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'app-release', version: '1.0.0'**

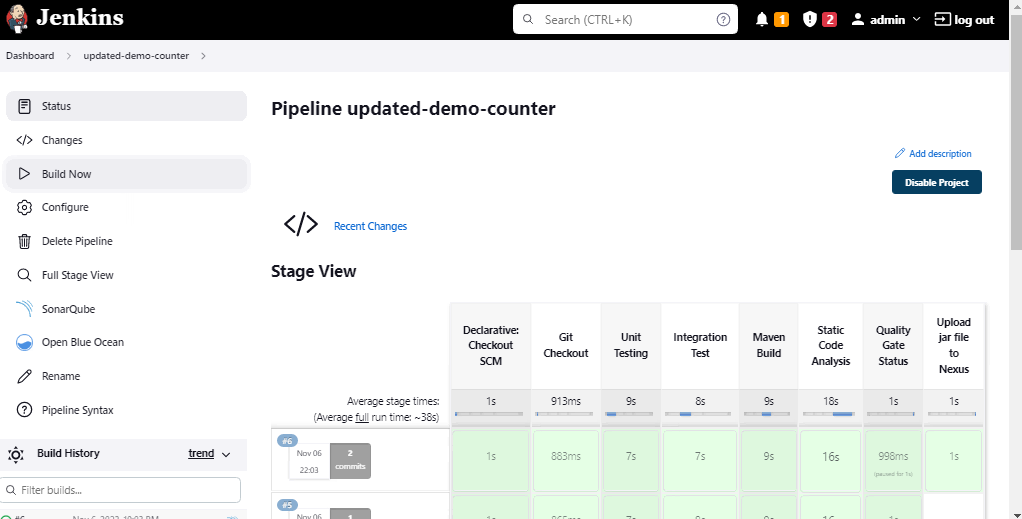
**Add Below Code to Jenkinsfile**

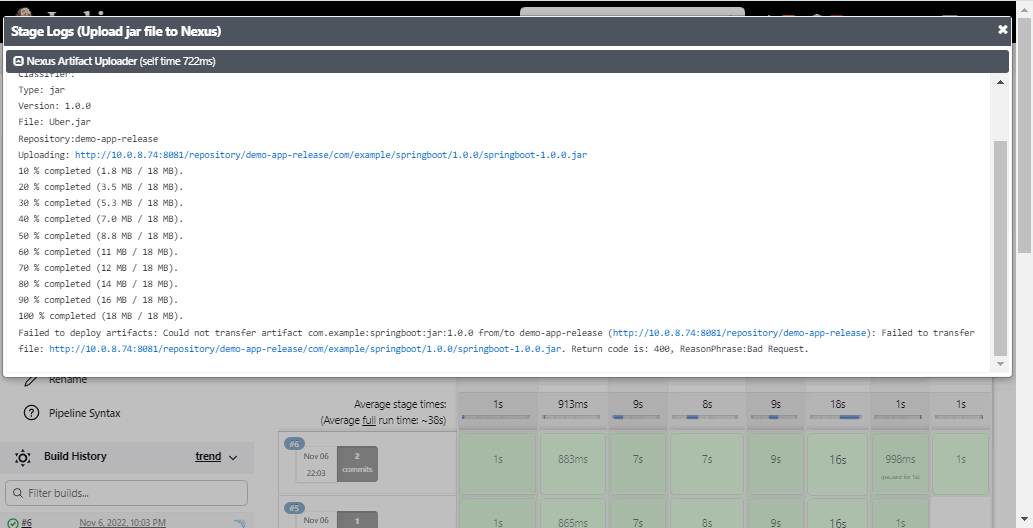
|  |
| --- |
| stage('Upload jar file to Nexus'){  steps{  script{  def readpomVersion = readMavenPom file: 'pom.xml'  nexusArtifactUploader artifacts:  [  [artifactId: 'springboot',  classifier: '',  file: 'target/Uber.jar',  type: 'jar']  ],  credentialsId: 'nexus-auth',  groupId: 'com.example',  nexusUrl: '10.0.8.74:8081',  nexusVersion: 'nexus3',  protocol: 'http',  repository: 'demo-app-release',  version: "${readpomVersion.version}"  }  }  } |

**Overall Updated Code**

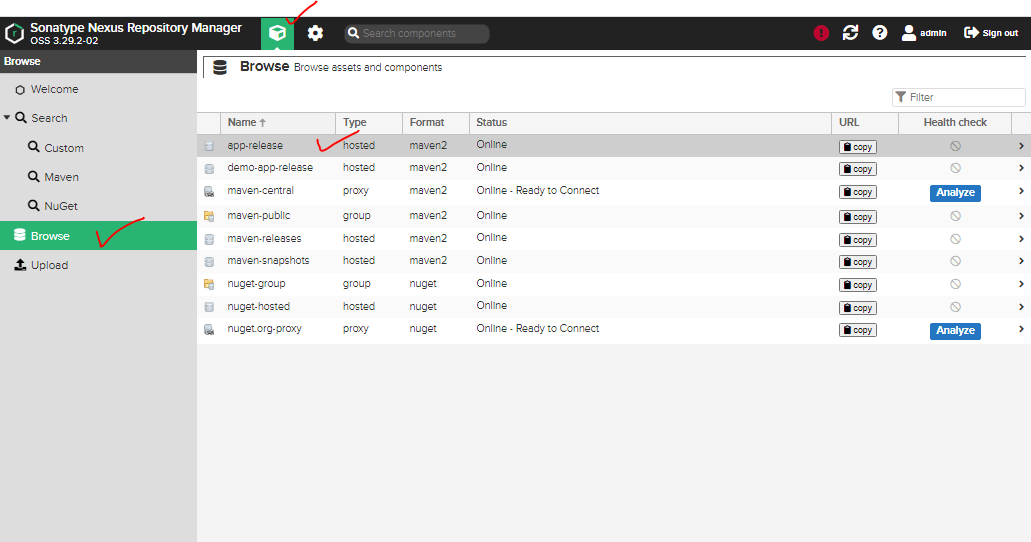
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | pipeline{ | |  |  | |  | agent any | |  | stages{ | |  | stage('Git Checkout'){ | |  | steps{ | |  | git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git' | |  | } | |  | } | |  | stage('Unit Testing'){ | |  | steps{ | |  | sh 'mvn test' | |  | } | |  | } | |  | stage('Integration Test'){ | |  | steps{ | |  | sh 'mvn verify -DskipUnitTests ' | |  | } | |  | } | |  | stage('Maven Build'){ | |  | steps{ | |  | sh 'mvn clean install' | |  | } | |  | } | |  | stage('Static Code Analysis'){ | |  | steps{ | |  | script{ | |  | withSonarQubeEnv(credentialsId: 'sonar-updated-api') { | |  | sh "mvn clean package sonar:sonar" | |  | } | |  | } | |  | } | |  | } | |  | stage('Quality Gate Status'){ | |  | steps{ | |  | script{ | |  | waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token' | |  | } | |  | } | |  | } | |  | stage('Upload jar file to Nexus'){ | |  | steps{ | |  | script{ | |  |  | |  | nexusArtifactUploader artifacts: | |  | [ | |  | [artifactId: 'springboot', | |  | classifier: '', | |  | file: 'target/Uber.jar', | |  | type: 'jar'] | |  | ], | |  |  | |  | credentialsId: 'nexus-auth', | |  | groupId: 'com.example', | |  | nexusUrl: '10.0.8.74:8081', | |  | nexusVersion: 'nexus3', | |  | protocol: 'http', | |  | repository: 'demo-app-release', | |  | version: "1.0.0” | |  |  | |  | } | |  | } | |  | } | |  | } | |  | } | |

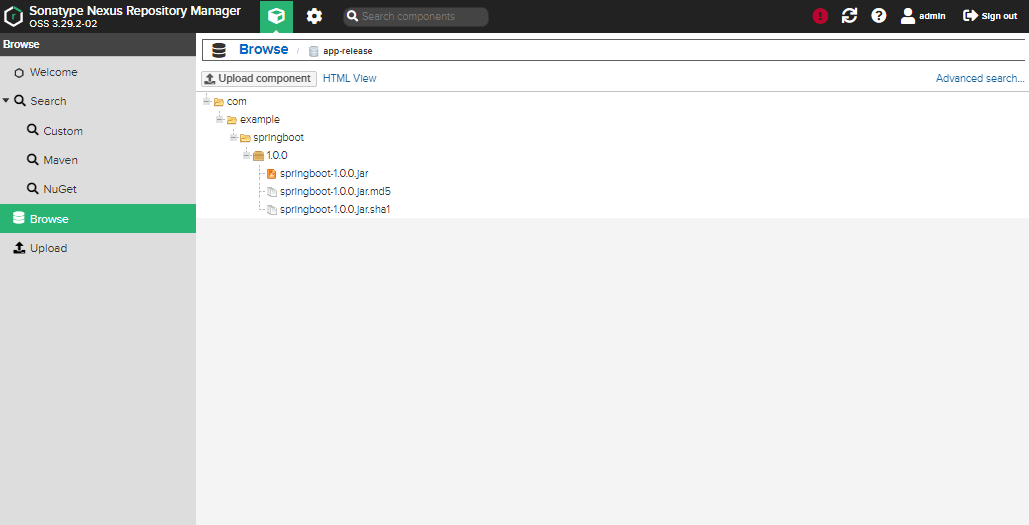
The stage got build successfully





Let us check if the Jar file got pushed to the Nexus Repo

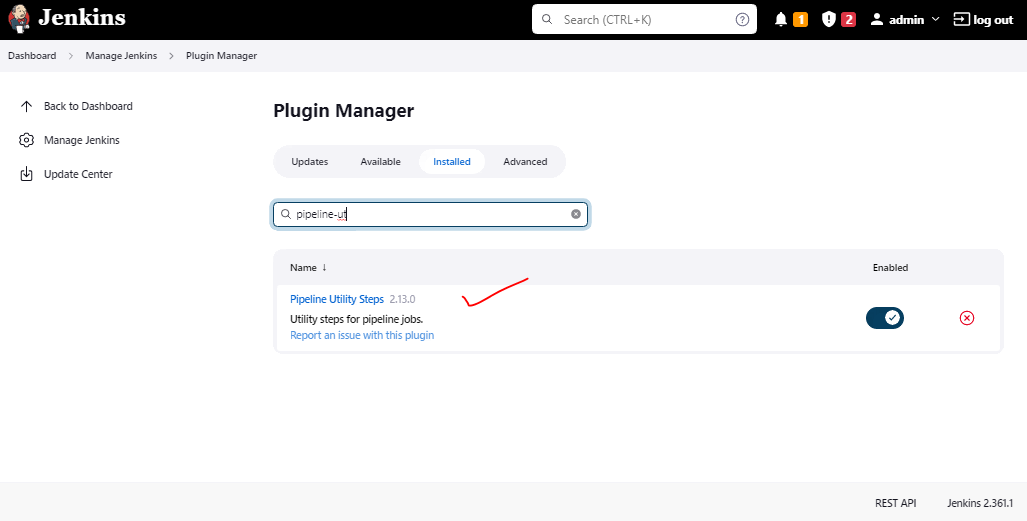




As we were hard coding the Version: It was taking Version: 1.0.0 – We will try to add some code and make it dynamic



**We need a Plugin for that. Add the below plugin in Jenkins**



Reading the Pipeline Utlity

https://www.jenkins.io/doc/pipeline/steps/pipeline-utility-steps/

readMavenPom: Read a maven project file

writeYaml: Write a yaml from an object or objects

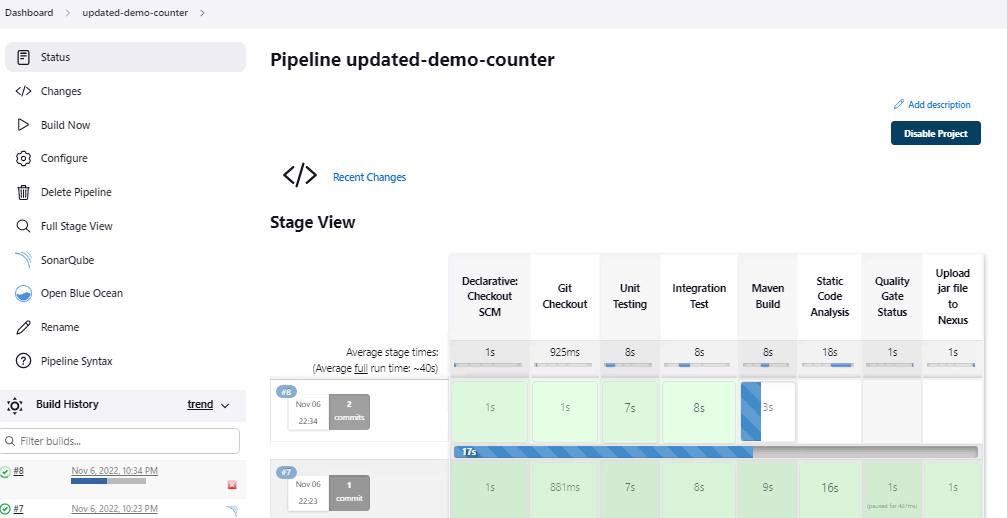
**Overall Updated Code**

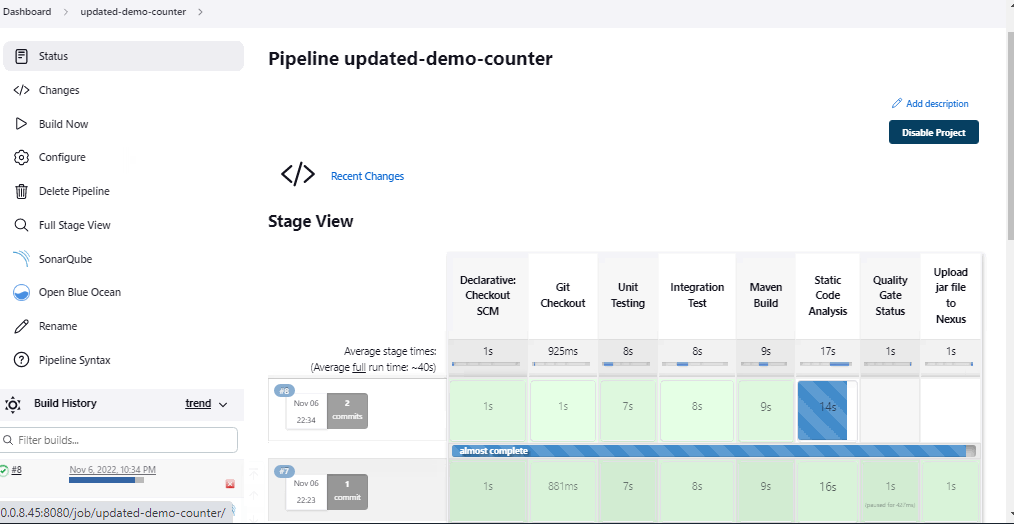
|  |
| --- |
| pipeline{  agent any  stages{  stage('Git Checkout'){  steps{  git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'  }  }  stage('Unit Testing'){  steps{  sh 'mvn test'  }  }  stage('Integration Test'){  steps{  sh 'mvn verify -DskipUnitTests '  }  }  stage('Maven Build'){  steps{  sh 'mvn clean install'  }  }  stage('Static Code Analysis'){  steps{  script{  withSonarQubeEnv(credentialsId: 'sonar-updated-api') {  sh "mvn clean package sonar:sonar"  }  }  }  }  stage('Quality Gate Status'){  steps{  script{  waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token'  }  }  }  stage('Upload jar file to Nexus'){  steps{  script{  def readpomVersion = readMavenPom file: 'pom.xml'  nexusArtifactUploader artifacts:  [  [artifactId: 'springboot',  classifier: '',  file: 'target/Uber.jar',  type: 'jar']  ],  credentialsId: 'nexus-auth',  groupId: 'com.example',  nexusUrl: '10.0.8.74:8081',  nexusVersion: 'nexus3',  protocol: 'http',  repository: 'app-release',  version: "${readpomVersion.version}"  }  }  }  }  } |

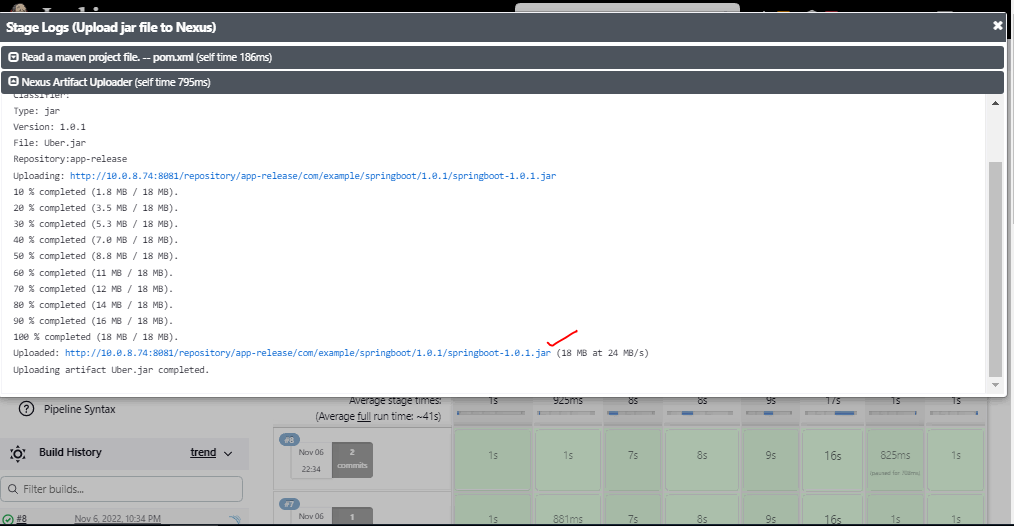
**Update the Pom.xml**

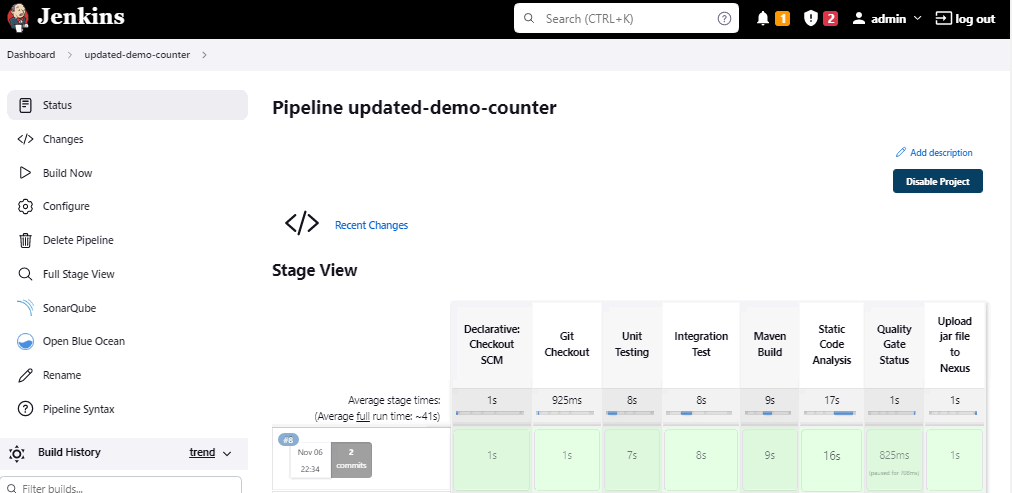


Run the Pipeline

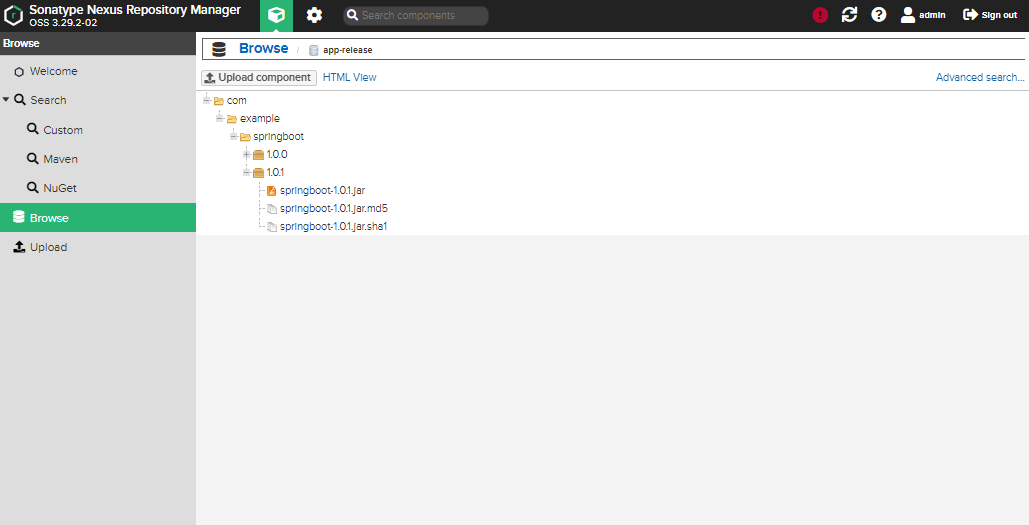




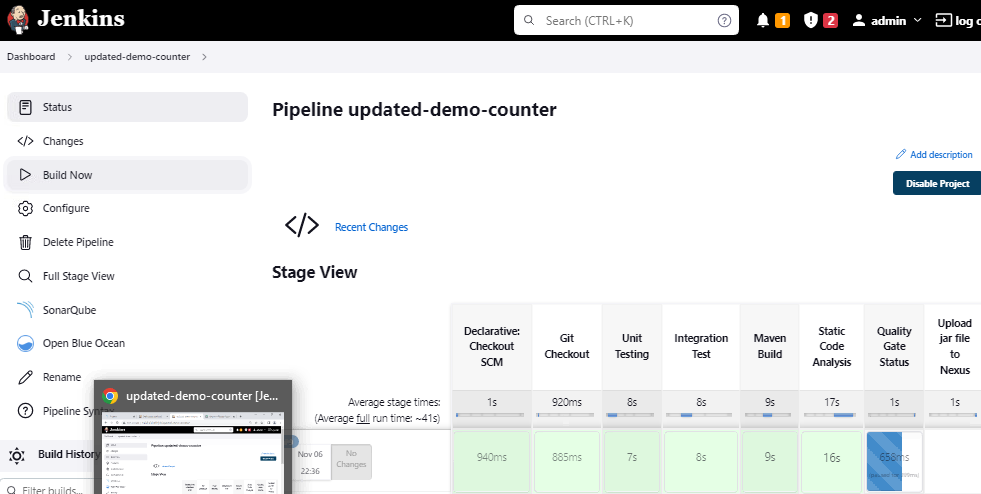




**The Nexus Repo is Updated**

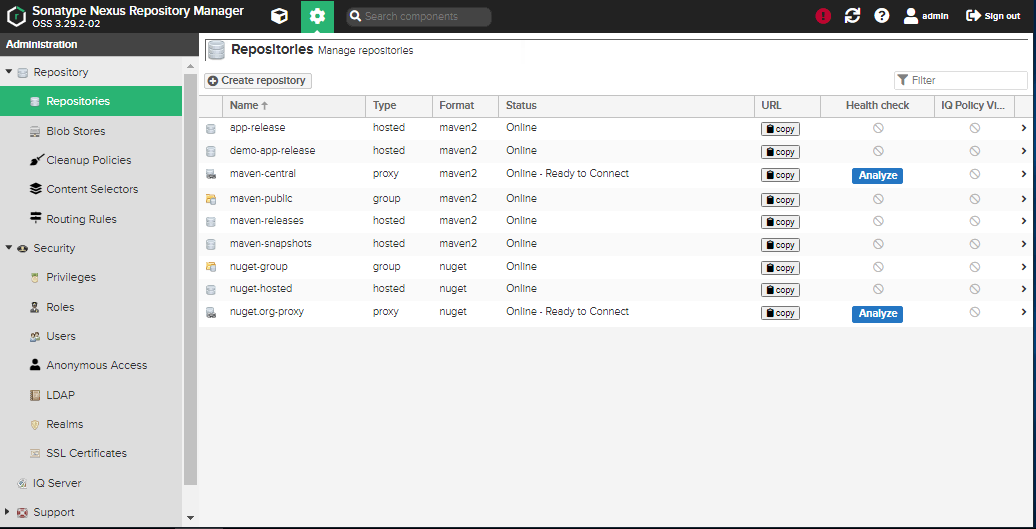


Rerun the Pipeline

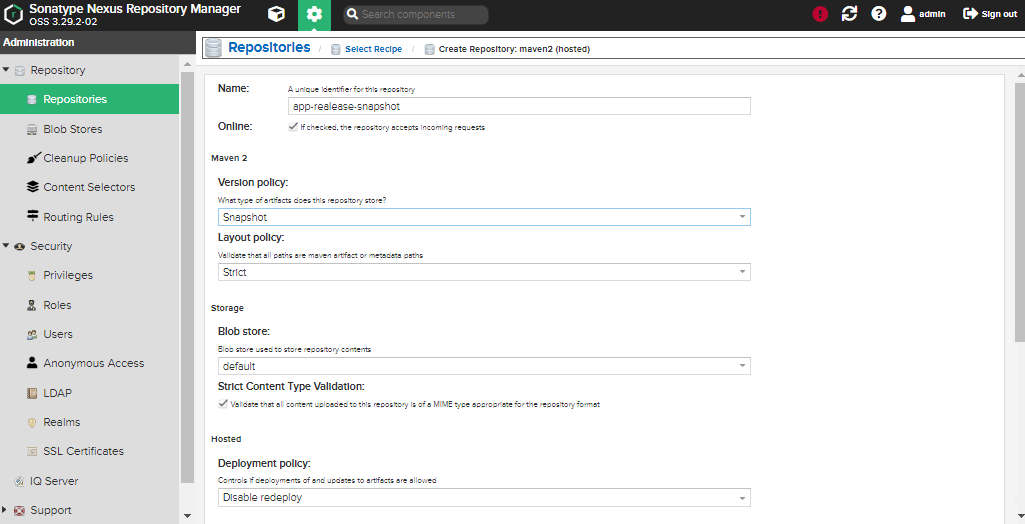


If we have to create the jar file and push it to Nexus as per the Latest & Snapshots. Let’s suppose we want to take the snapshot jar and upload it to Nexus. We have to mentioned it in Jenkinsfile stating that either you take the latest artifacts or the other which contains “SNAPSHOT” word. To make this happen we have to use mavenReadFile in the Jenkinsfile.

We will update the POM.XML and also the Jenkinsfile, and check if it takes. Here is the updated POM.xml and Jenkinsfile



Create a New Repository for Snapshot



**Add this Code**:

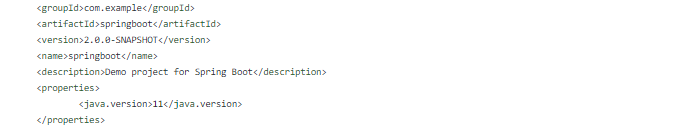
def nexusRepo = readpomVersion .version.endsWith('SNAPSHOT') ? "app-realease-snapshot" : "app-release"

repository: nexusRepo,

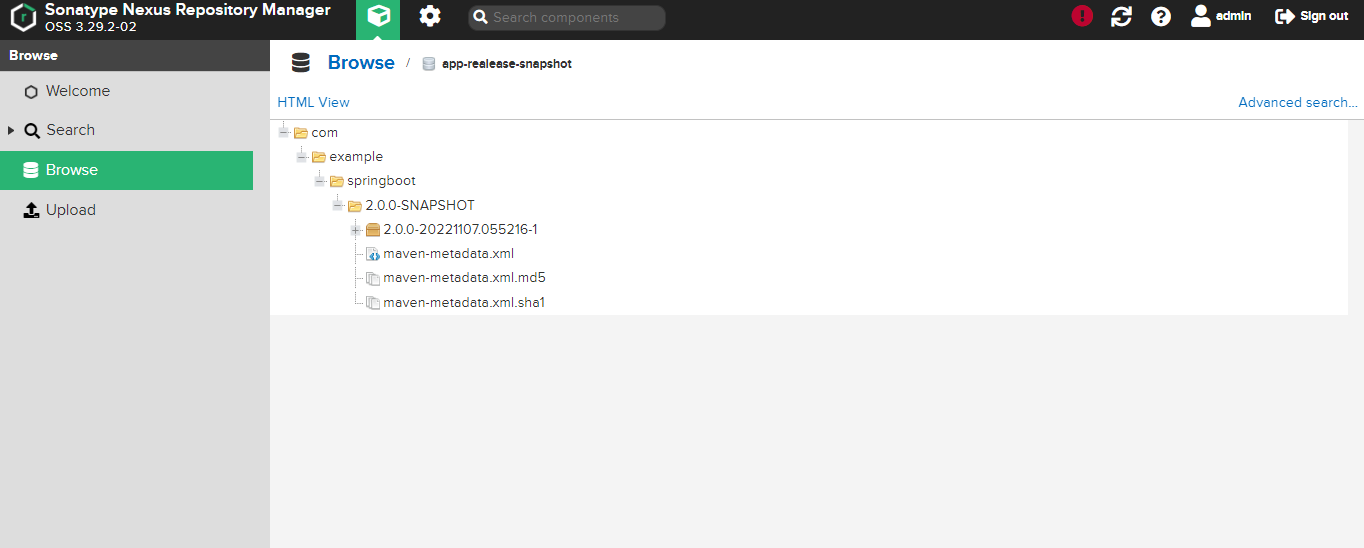
**Jenkins File Updated**



**Updated the POM.XML**



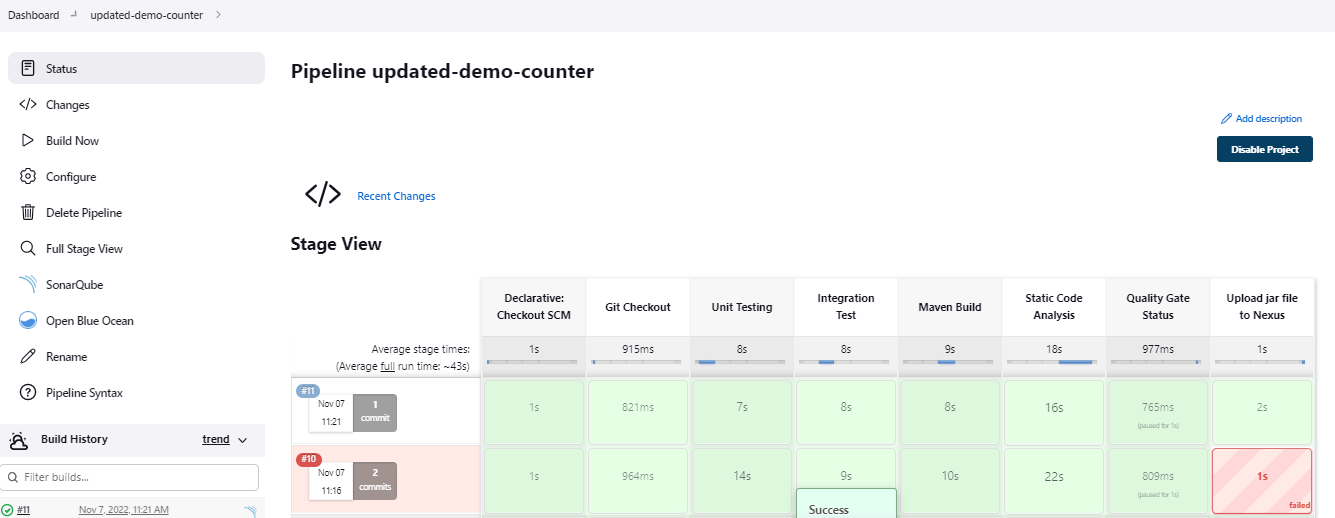
**It is created the Jar file in the New Repository**



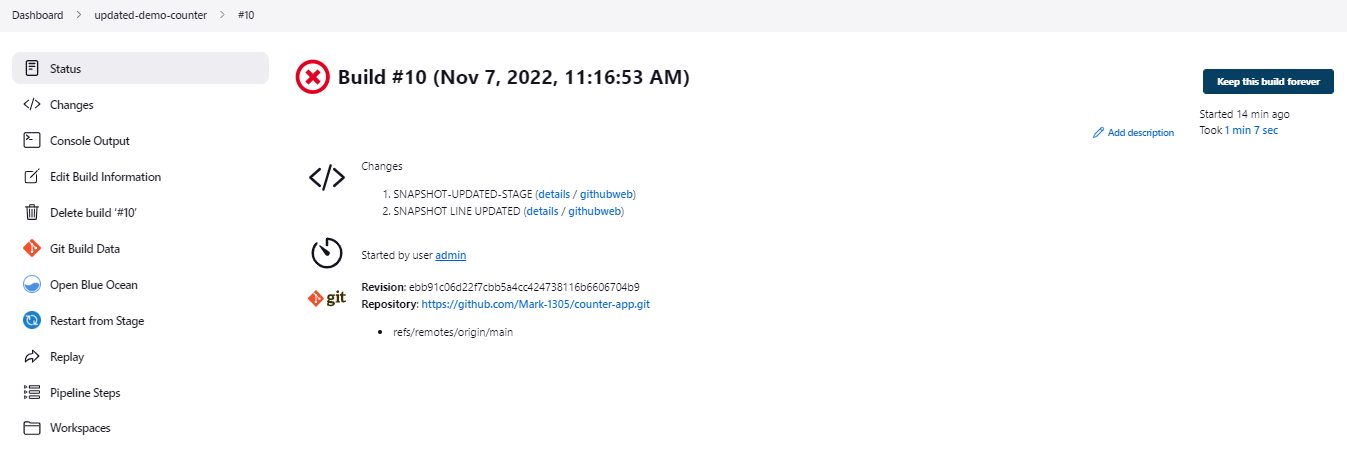
**The Overall Code**

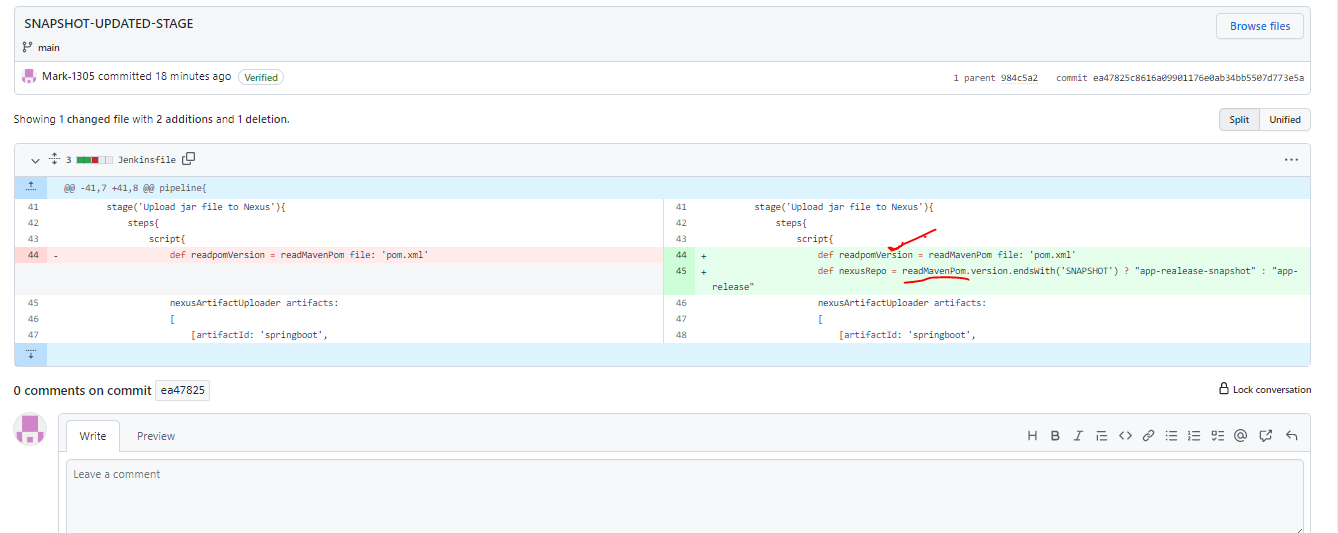
|  |
| --- |
| pipeline{  agent any  stages{  stage('Git Checkout'){  steps{  git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'  }  }  stage('Unit Testing'){  steps{  sh 'mvn test'  }  }  stage('Integration Test'){  steps{  sh 'mvn verify -DskipUnitTests '  }  }  stage('Maven Build'){  steps{  sh 'mvn clean install'  }  }  stage('Static Code Analysis'){  steps{  script{  withSonarQubeEnv(credentialsId: 'sonar-updated-api') {  sh "mvn clean package sonar:sonar"  }  }  }  }  stage('Quality Gate Status'){  steps{  script{  waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token'  }  }  }  stage('Upload jar file to Nexus'){  steps{  script{  def readpomVersion = readMavenPom file: 'pom.xml'  def nexusRepo = readpomVersion .version.endsWith('SNAPSHOT') ? "app-realease-snapshot" : "app-release"  nexusArtifactUploader artifacts:  [  [artifactId: 'springboot',  classifier: '',  file: 'target/Uber.jar',  type: 'jar']  ],  credentialsId: 'nexus-auth',  groupId: 'com.example',  nexusUrl: '10.0.8.74:8081',  nexusVersion: 'nexus3',  protocol: 'http',  repository: nexusRepo,  version: "${readpomVersion.version}"  }  }  }  }  } |

Built run Successfully

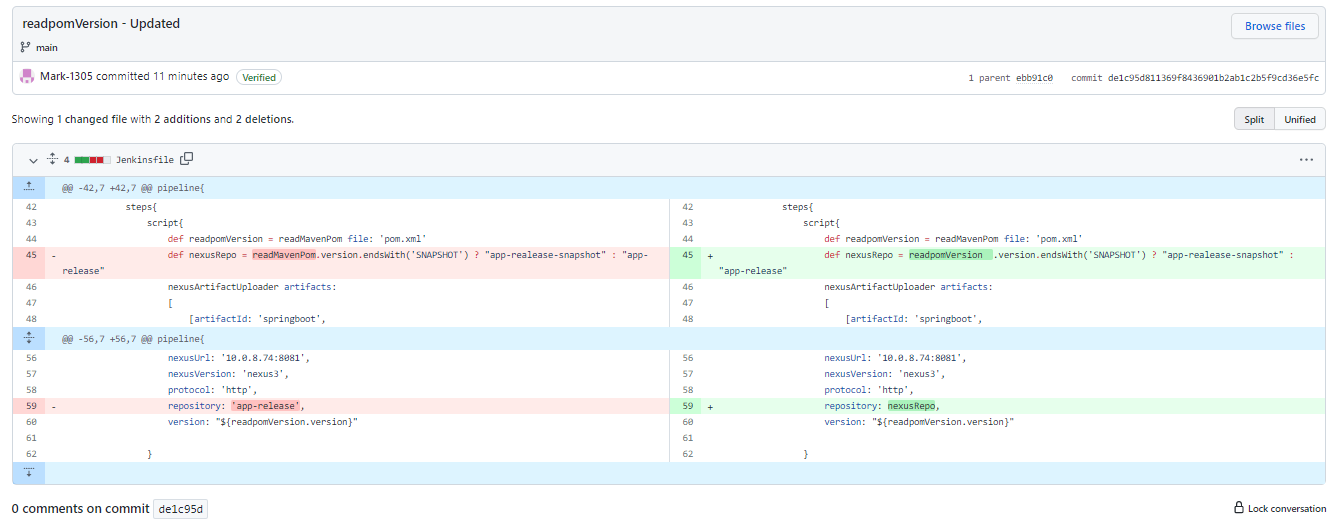


That Built failed earlier as there was some error with mavenRead syantx



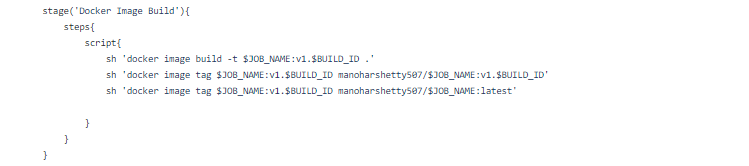


**The Correct One**



**Building Docker – To Deploy the Application**

Update the Jenkinsfile



**The overall Code**

|  |
| --- |
| pipeline{  agent any  stages{  stage('Git Checkout'){  steps{  git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'  }  }  stage('Unit Testing'){  steps{  sh 'mvn test'  }  }  stage('Integration Test'){  steps{  sh 'mvn verify -DskipUnitTests '  }  }  stage('Maven Build'){  steps{  sh 'mvn clean install'  }  }  stage('Static Code Analysis'){  steps{  script{  withSonarQubeEnv(credentialsId: 'sonar-updated-api') {  sh "mvn clean package sonar:sonar"  }  }  }  }  stage('Quality Gate Status'){  steps{  script{  waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token'  }  }  }  stage('Upload jar file to Nexus'){  steps{  script{  def readpomVersion = readMavenPom file: 'pom.xml'  def nexusRepo = readpomVersion .version.endsWith('SNAPSHOT') ? "app-realease-snapshot" : "app-release"  nexusArtifactUploader artifacts:  [  [artifactId: 'springboot',  classifier: '',  file: 'target/Uber.jar',  type: 'jar']  ],  credentialsId: 'nexus-auth',  groupId: 'com.example',  nexusUrl: '10.0.8.74:8081',  nexusVersion: 'nexus3',  protocol: 'http',  repository: nexusRepo,  version: "${readpomVersion.version}"  }  }  }  stage('Docker Image Build'){  steps{  script{  sh 'docker image build -t $JOB\_NAME:v1.$BUILD\_ID .'  sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:v1.$BUILD\_ID'  sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:latest'    }  }  }  }  } |

The Build failed



Forgot to Add (.) in the build line

|  |
| --- |
| stage('Docker Image Build'){  steps{  script{  sh 'docker image build -t $JOB\_NAME:v1.$BUILD\_ID '  sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:v1.$BUILD\_ID'  sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:latest'    }  }  } |

**Added it**

stage('Docker Image Build'){

steps{

script{

sh 'docker image build -t $JOB\_NAME:v1.$BUILD\_ID .'

sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:v1.$BUILD\_ID'

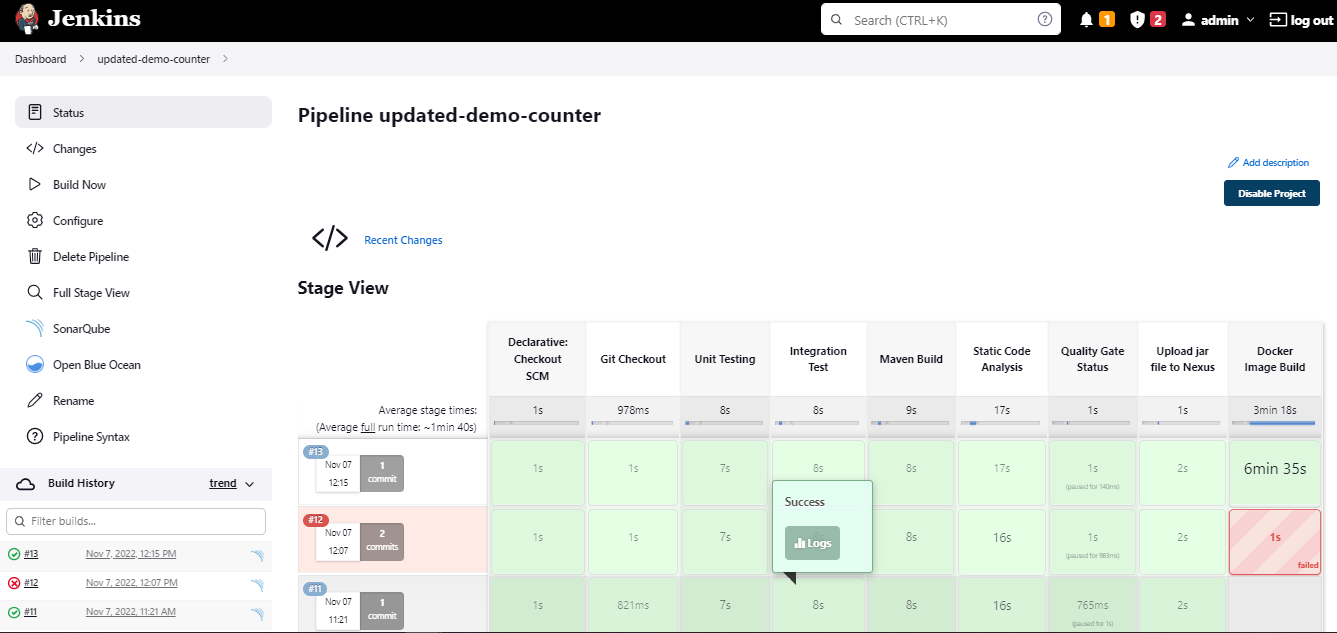
sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:latest'

}

}

}

It got Build Now – Docker Image is created and it is tagged with Latest and also with the Job\_Name and Build ID

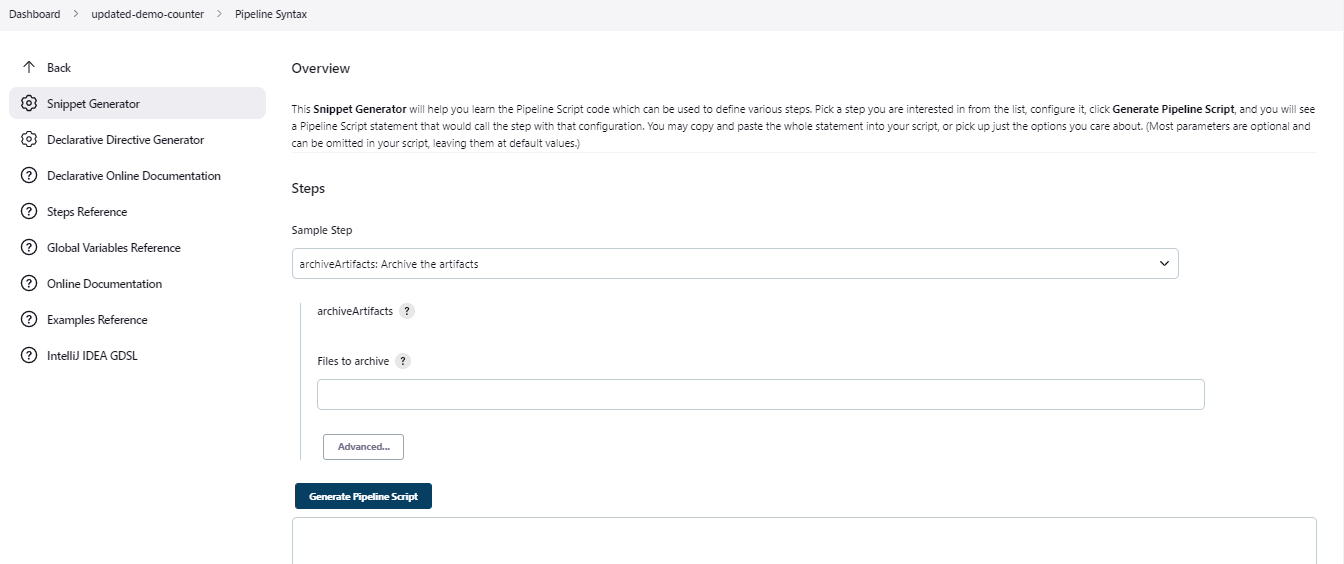


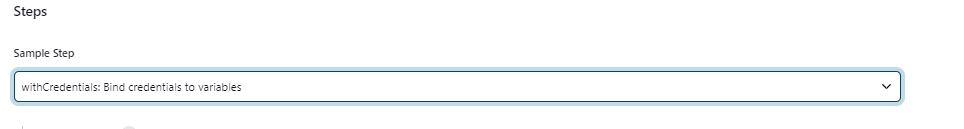
The Built Image



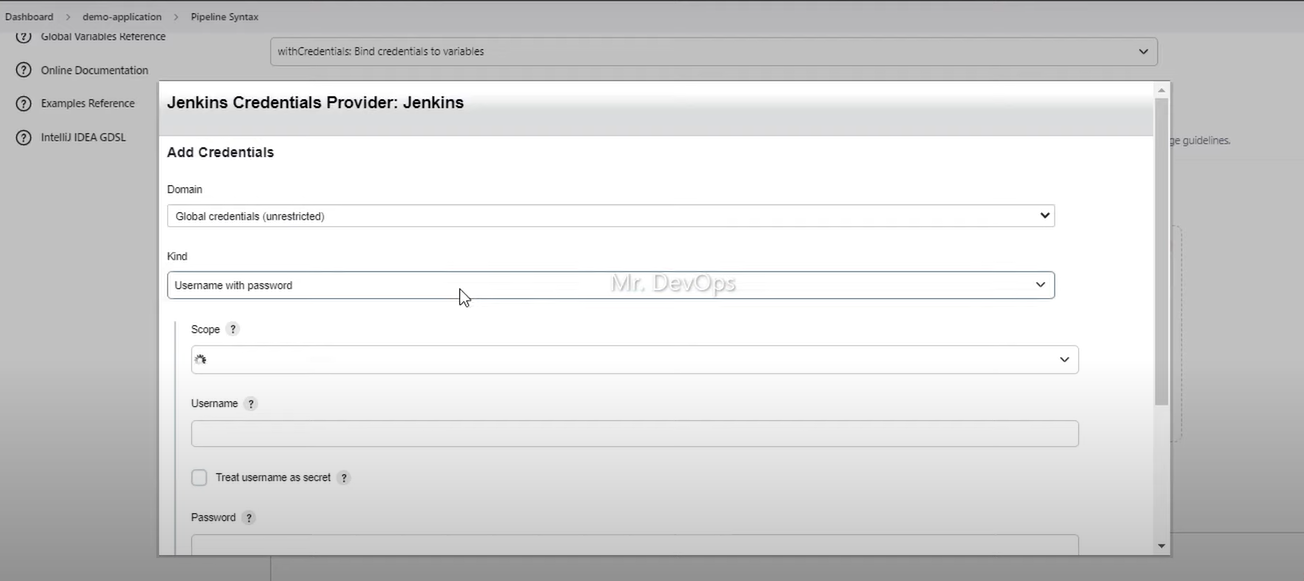
Pushing Docker Image to Docker HUB

This requires us to login into Docker Hub

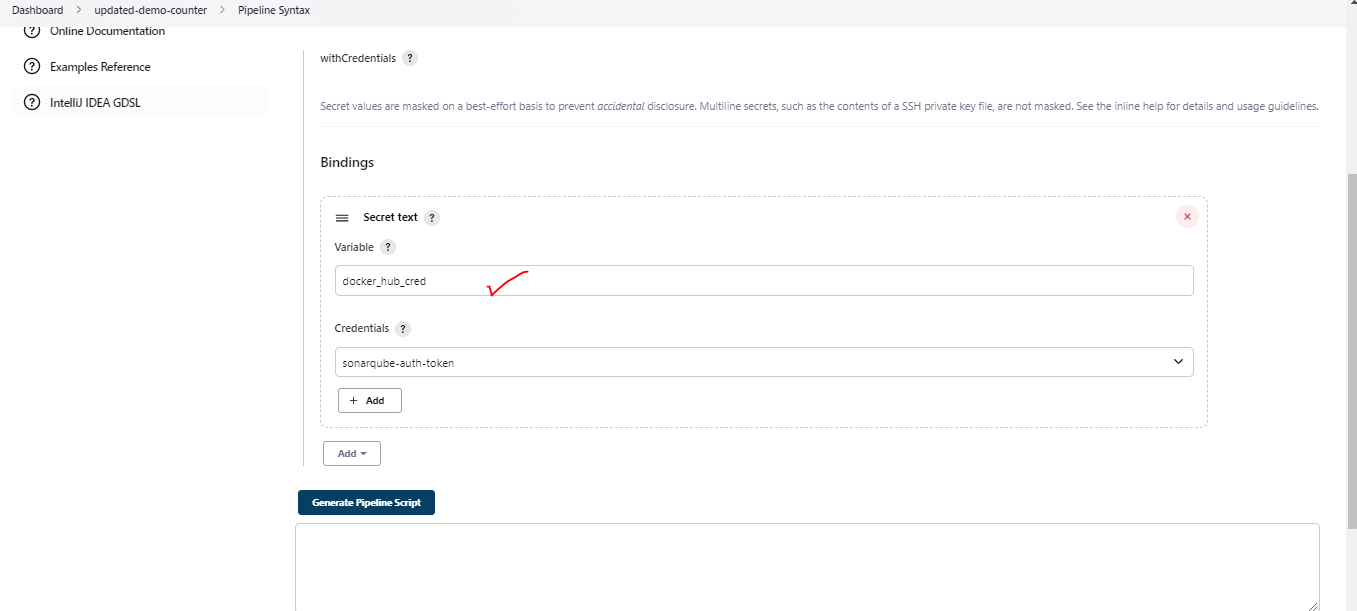


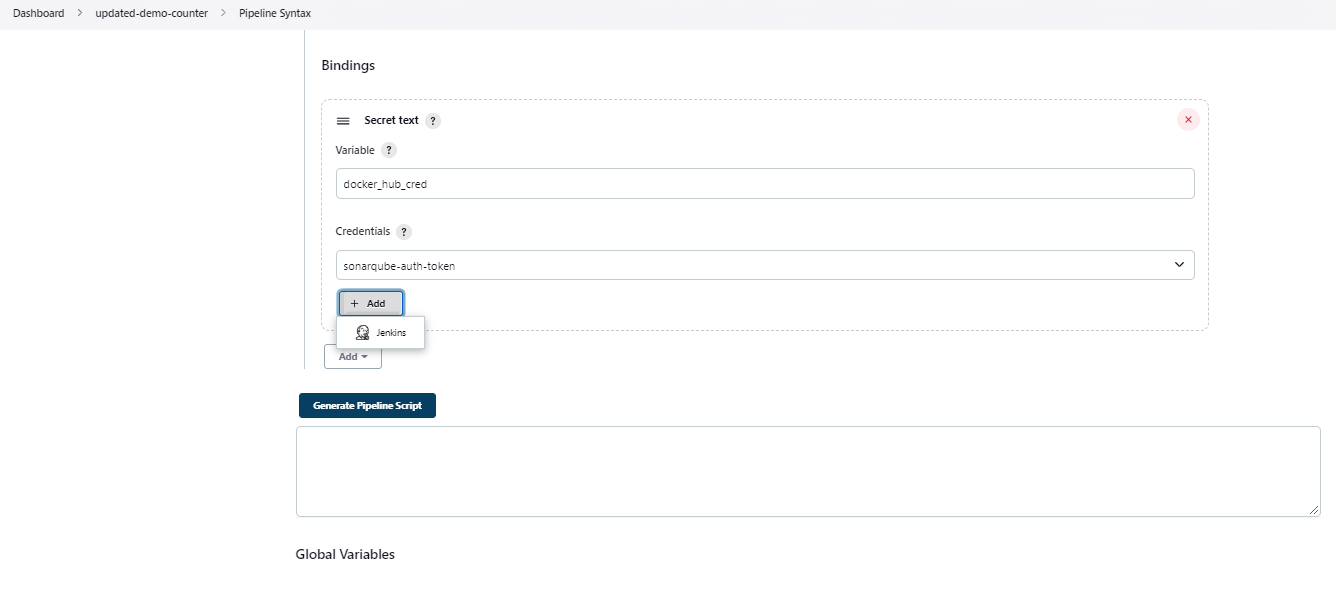
****

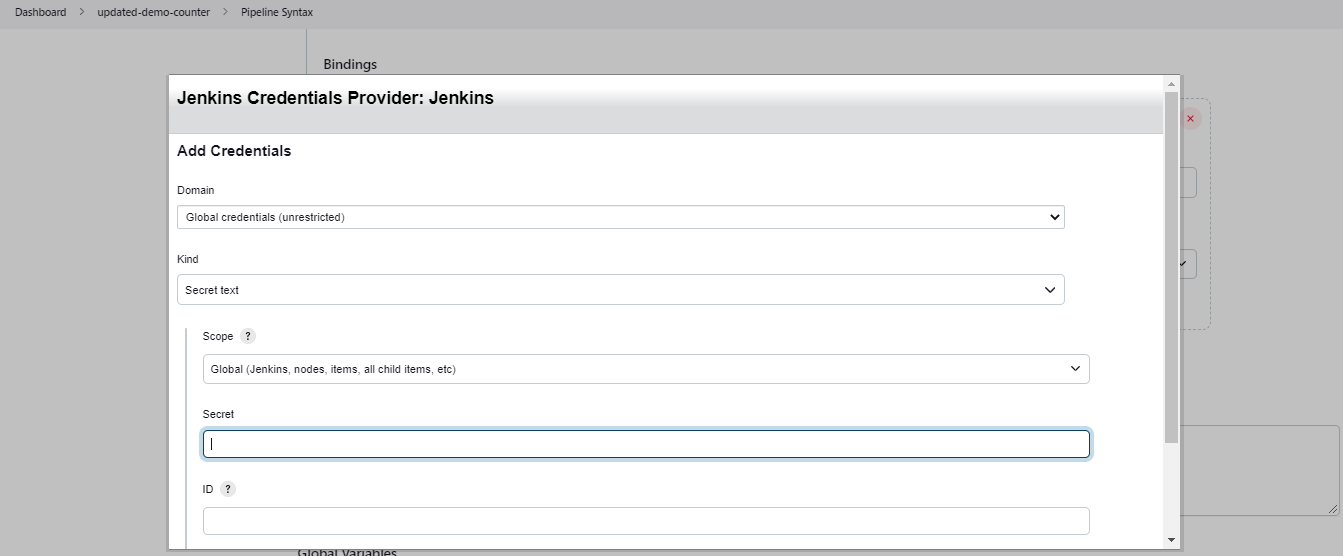
**Create a Secret Text and Store DockerHub Password**

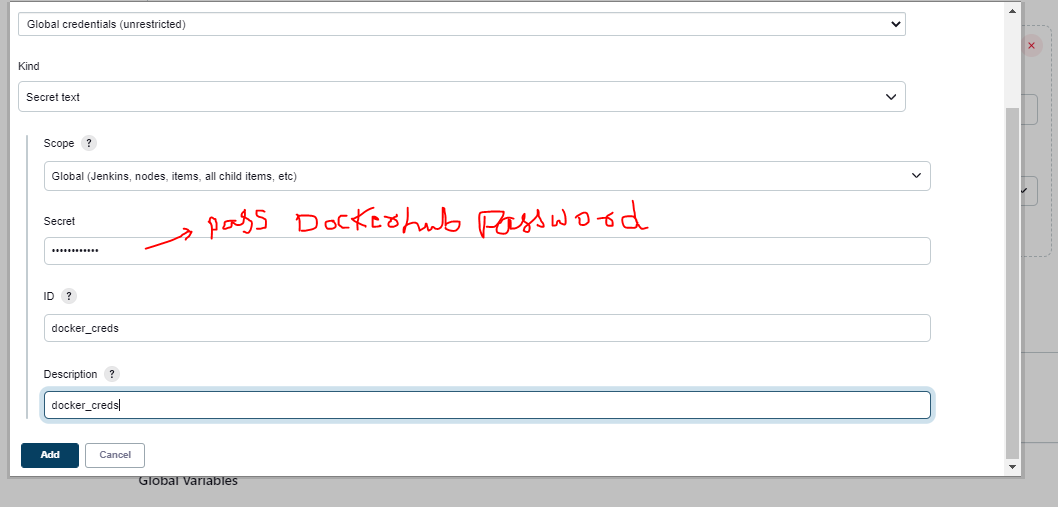












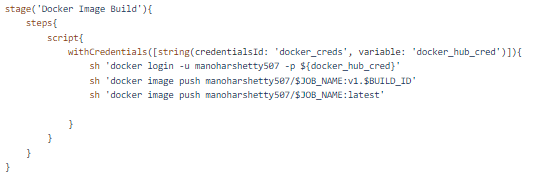
**Generated Syntax**

withCredentials([string(credentialsId: 'docker\_creds', variable: 'docker\_hub\_cred')]) {

// some block

}

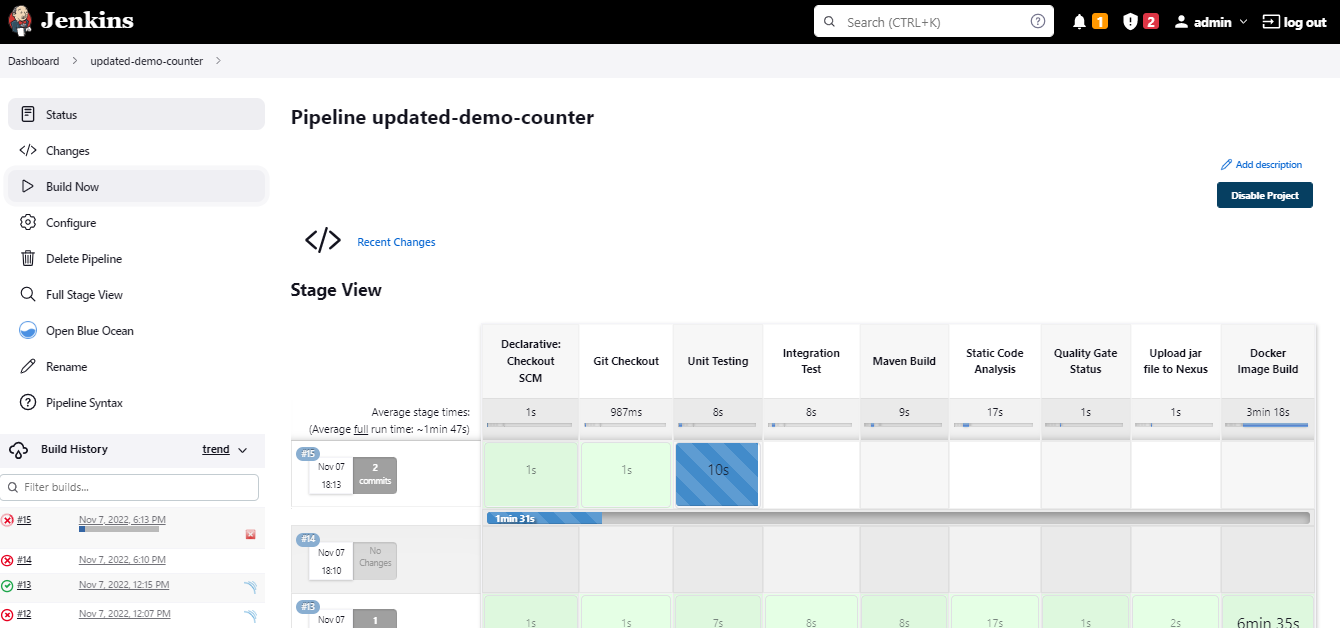
**Add the below line**

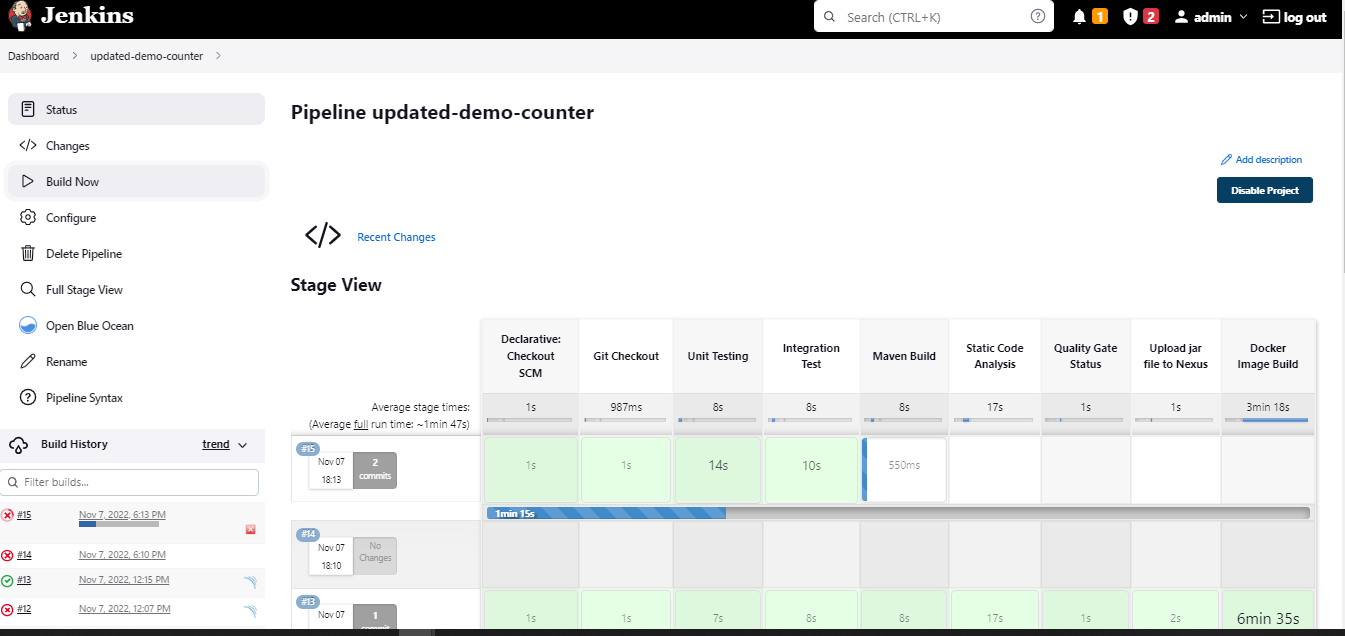


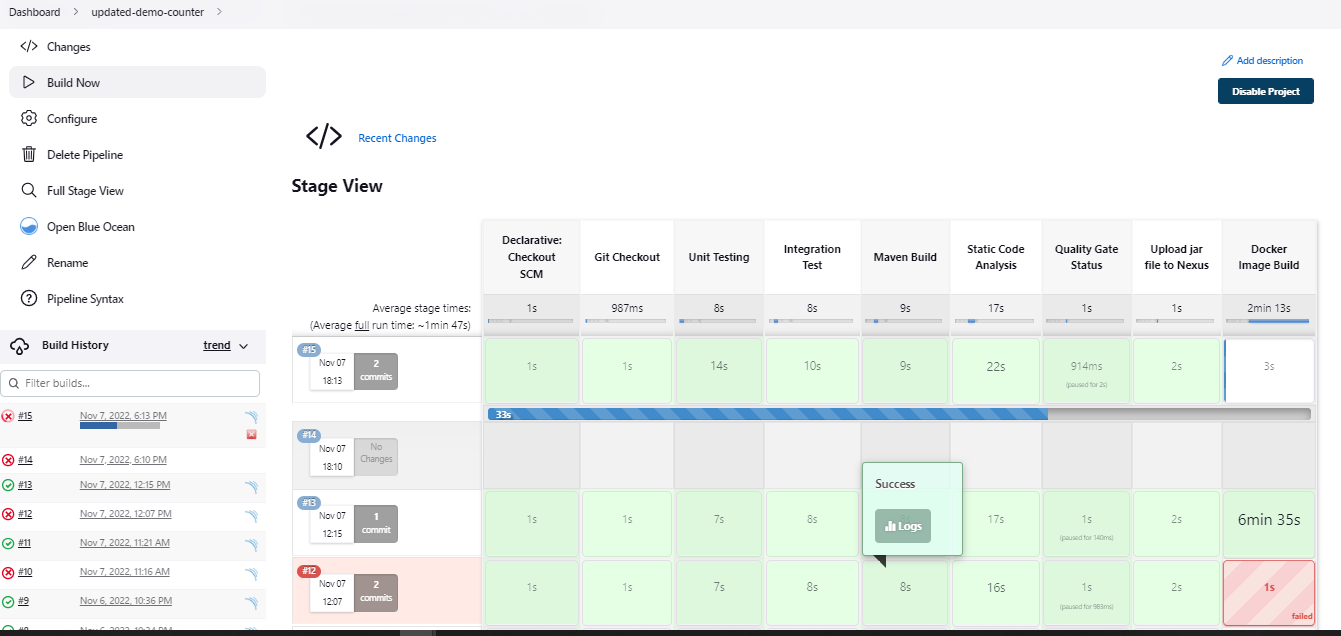
The Overall Code

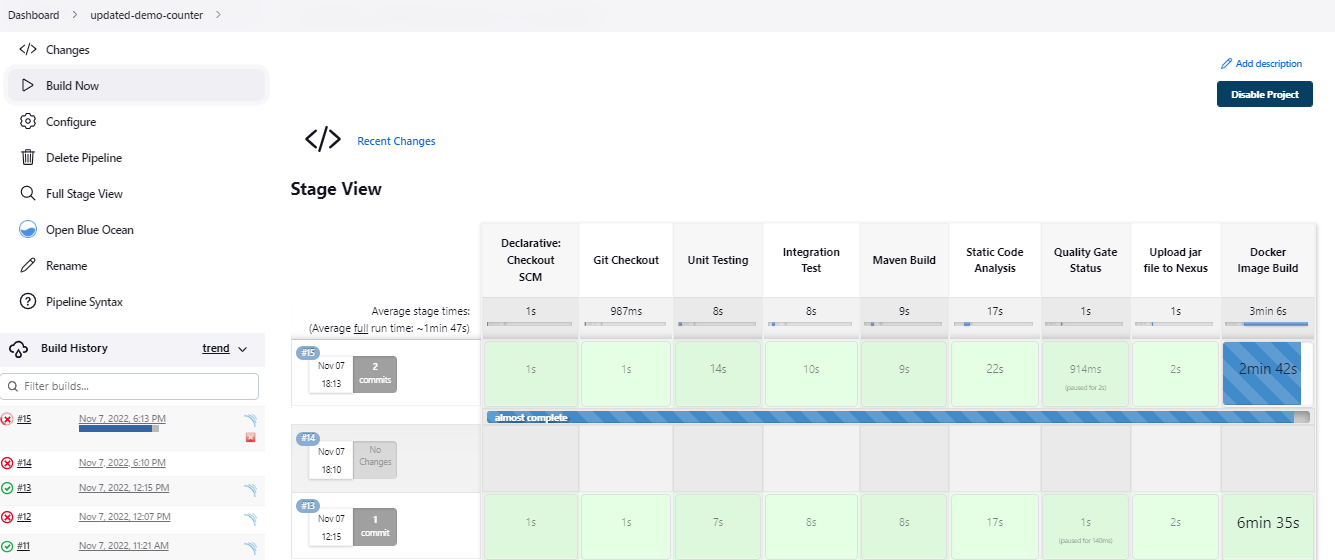
|  |
| --- |
| pipeline{  agent any  stages{  stage('Git Checkout'){  steps{  git branch: 'main', url: 'https://github.com/Mark-1305/counter-app.git'  }  }  stage('Unit Testing'){  steps{  sh 'mvn test'  }  }  stage('Integration Test'){  steps{  sh 'mvn verify -DskipUnitTests '  }  }  stage('Maven Build'){  steps{  sh 'mvn clean install'  }  }  stage('Static Code Analysis'){  steps{  script{  withSonarQubeEnv(credentialsId: 'sonar-updated-api') {  sh "mvn clean package sonar:sonar"  }  }  }  }  stage('Quality Gate Status'){  steps{  script{  waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-auth-token'  }  }  }  stage('Upload jar file to Nexus'){  steps{  script{  def readpomVersion = readMavenPom file: 'pom.xml'  def nexusRepo = readpomVersion .version.endsWith('SNAPSHOT') ? "app-realease-snapshot" : "app-release"  nexusArtifactUploader artifacts:  [  [artifactId: 'springboot',  classifier: '',  file: 'target/Uber.jar',  type: 'jar']  ],  credentialsId: 'nexus-auth',  groupId: 'com.example',  nexusUrl: '10.0.8.74:8081',  nexusVersion: 'nexus3',  protocol: 'http',  repository: nexusRepo,  version: "${readpomVersion.version}"  }  }  }  stage('Docker Image Build'){  steps{  script{  sh 'docker image build -t $JOB\_NAME:v1.$BUILD\_ID .'  sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:v1.$BUILD\_ID'  sh 'docker image tag $JOB\_NAME:v1.$BUILD\_ID manoharshetty507/$JOB\_NAME:latest'    }  }  }  stage('Docker Image Build'){  steps{  script{  withCredentials([string(credentialsId: 'docker\_creds', variable: 'docker\_hub\_cred')]){  sh 'docker login -u manoharshetty507 -p ${docker\_hub\_cred}'  sh 'docker image push manoharshetty507/$JOB\_NAME:v1.$BUILD\_ID'  sh 'docker image push manoharshetty507/$JOB\_NAME:latest'    }  }  }  }  }  } |

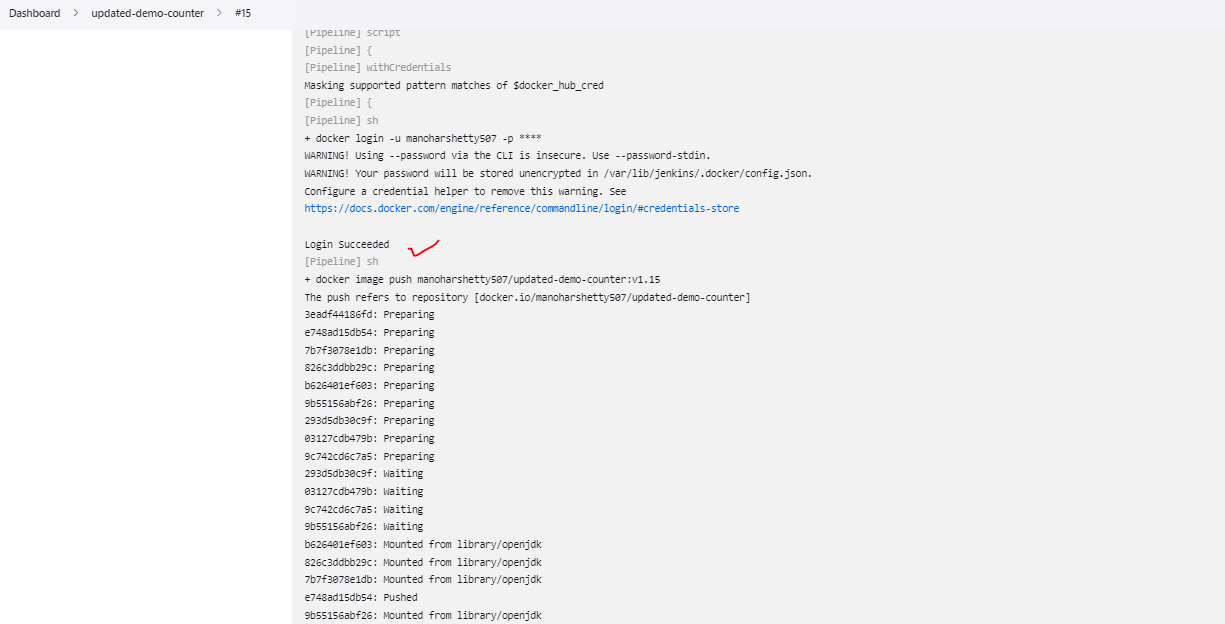
Build in Process



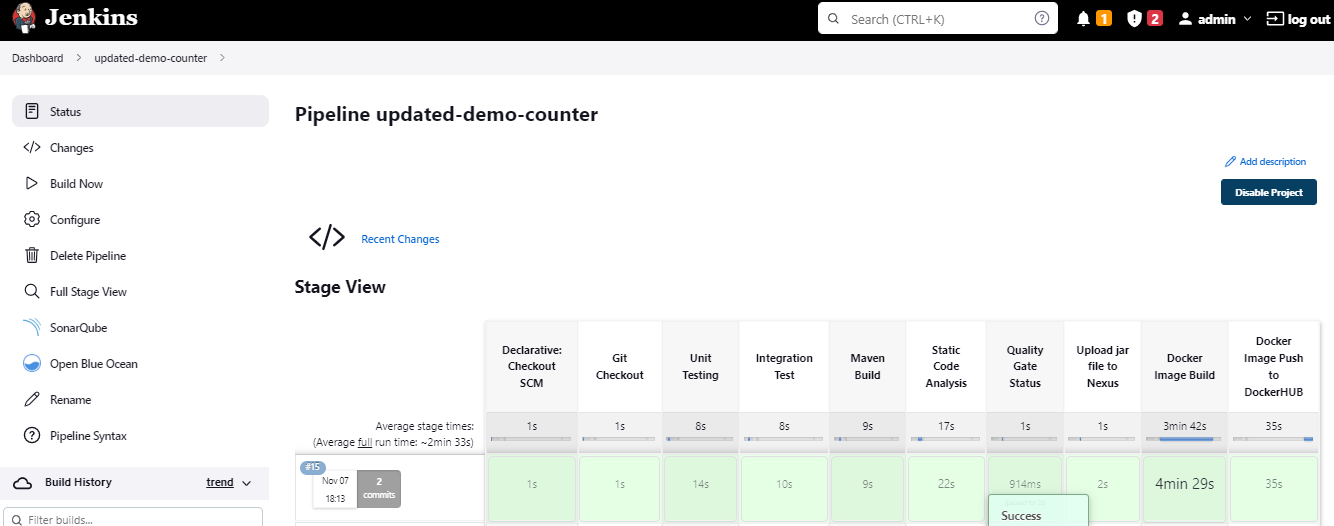








**The Build Succeeded – Image got Pushed to Docker Hub**



**The Image got Updated in the DockerHub**

