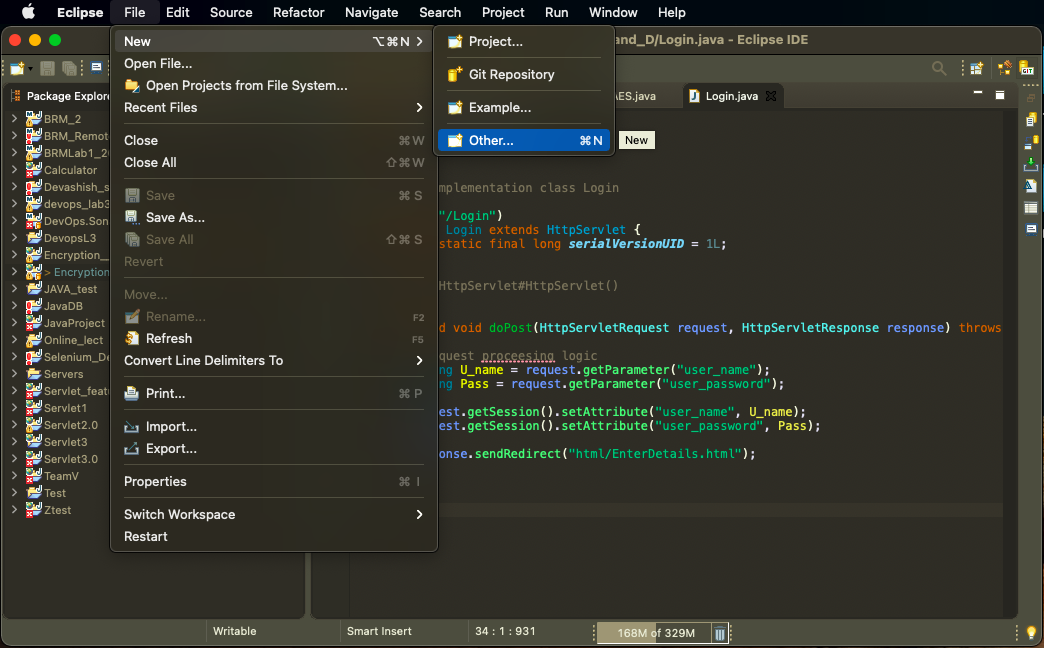
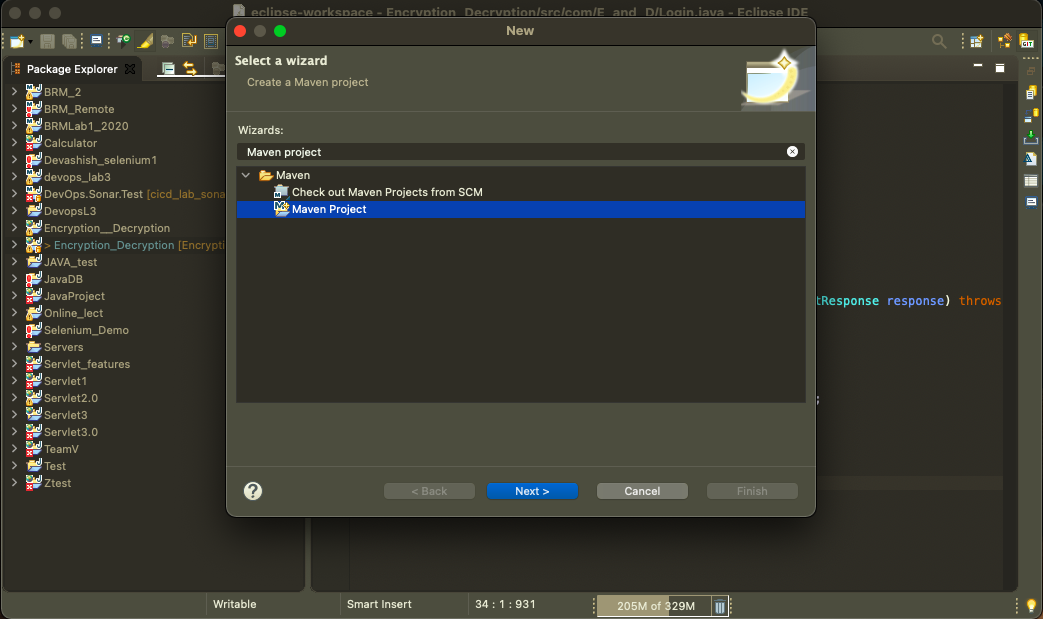
***Create step by step a simple maven project in Eclipse IDE.***

**Step-1**

* Open Eclipse
* Click on *File* -> *New* -> *other* ->*Maven Project*

**

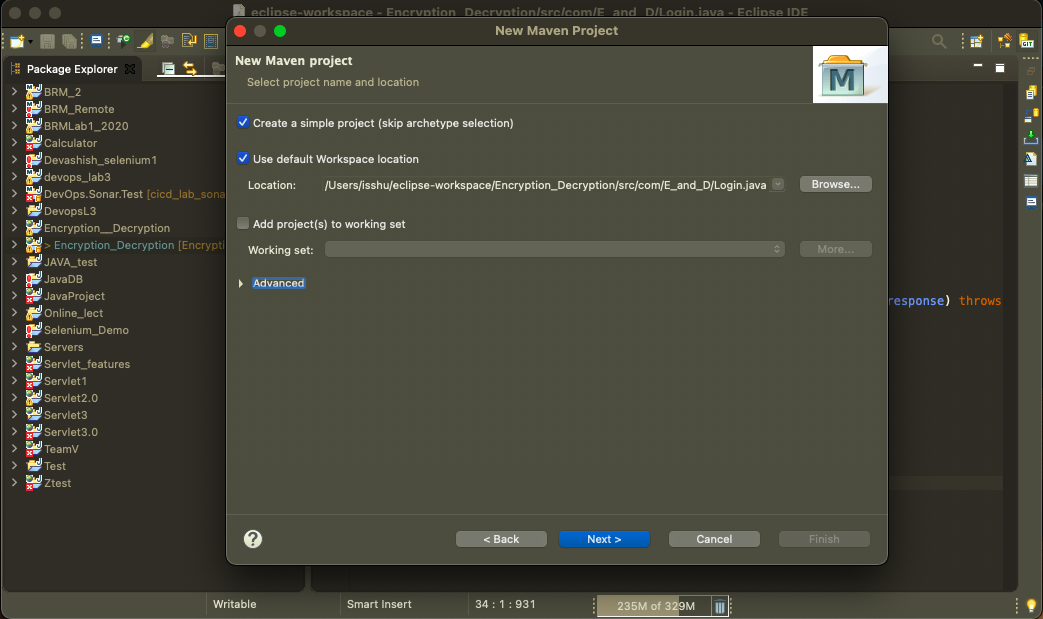
**

**Step-2**

Click on Checkbox for both

* Create a simple project
* Use default Workspace location
* Click on next button

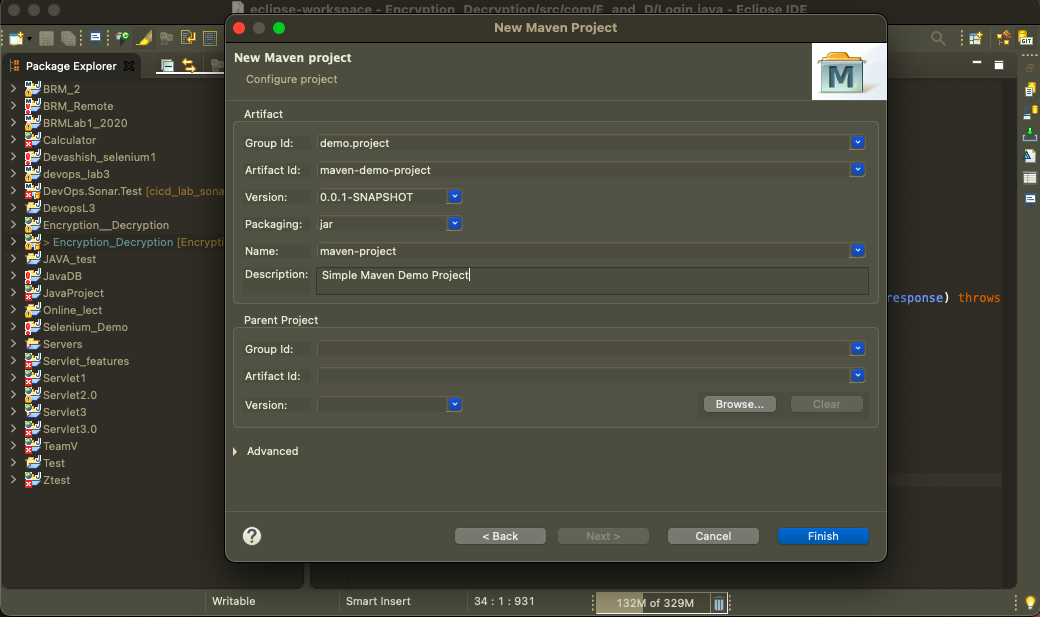
Note:- Check the option “Create a simple project (skip archetype selection)”.



**Step-3**

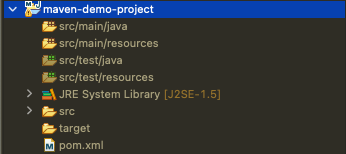
Provide GroupId and ArtifactId in next screen.

* **GroupId**: demo.project
* **Artifact Id**: maven-demo-project
* **Name**: maven-project
* **Description**: Simple Maven Demo Project



**Step-4**

And you are all set. You should see a new Project in Eclipse with below structure.

**

## Step-5

As you can see in the maven project structure, the default java compiler version ( i.e. source and target setting ) is 1.5. To change the default settings, add the following snippet to**pom.xml**.

<**build**>

<**plugins**>

<**plugin**>

<**groupId**>org.apache.maven.plugins</**groupId**>

<**artifactId**>maven-compiler-plugin</**artifactId**>

<**version**>3.6.0</**version**>

<**configuration**>

<**source**>1.8</**source**>

<**target**>1.8</**target**>

</**configuration**>

</**plugin**>

</**plugins**>

</**build**>

After changes in **pom.xml** file, update the maven project. To update maven project right click

on  *maven-project* → *Maven* → *Update Project*.

## Step-6

Add some dependencies to **pom.xml** file. Here we are adding *junit* dependency to **pom.xml.**

<**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 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<**modelVersion**>4.0.0</**modelVersion**>

<**groupId**>demo.project</**groupId**>

<**artifactId**>maven-demo-project</**artifactId**>

<**version**>0.0.1-SNAPSHOT</**version**>

<**name**>maven-project</**name**>

<**description**>Simple Maven Demo Project</**description**>

<**dependencies**>

<**dependency**>

<**groupId**>junit</**groupId**>

<**artifactId**>junit</**artifactId**>

<**version**>4.12</**version**>

</**dependency**>

</**dependencies**>

<**build**>

<**plugins**>

<**plugin**>

<**groupId**>org.apache.maven.plugins</**groupId**>

<**artifactId**>maven-compiler-plugin</**artifactId**>

<**version**>3.6.0</**version**>

<**configuration**>

<**source**>1.8</**source**>

<**target**>1.8</**target**>

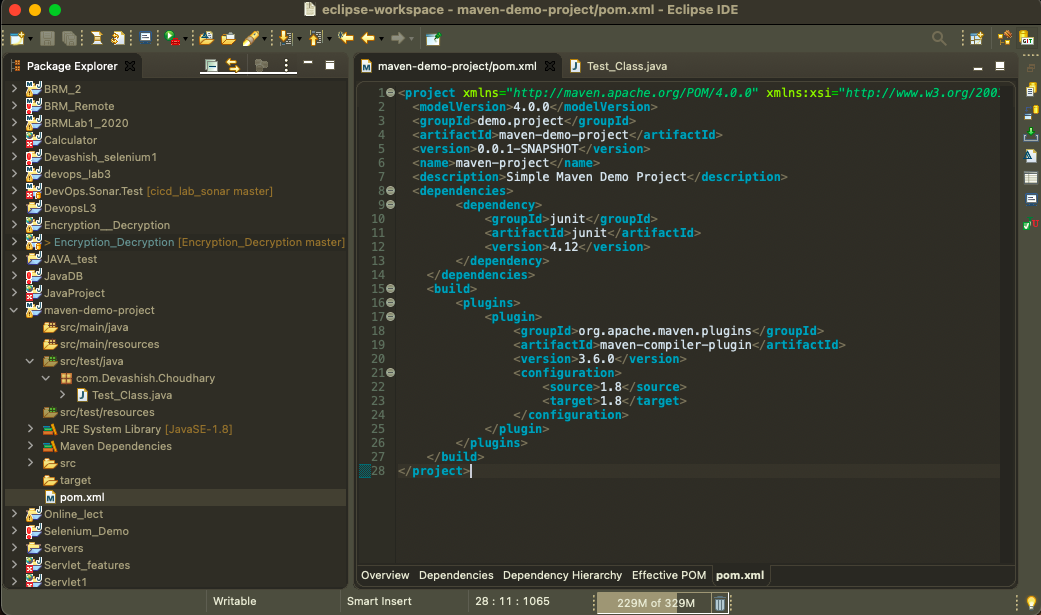
</**configuration**>

</**plugin**>

</**plugins**>

</**build**>

</**project**>



Let's test our created project by creating a simple JUnit test.

**Step-7**

Create a package, named as *com.Devashish.Choudhary*, under *src/test/java* folder. Now create a class *Test\_class.java* under *src/test/java* package and write the following code in it.

package com.Devashish.Choudhary;

import org.junit.Assert;

import org.junit.Test;

public class Test\_Class {

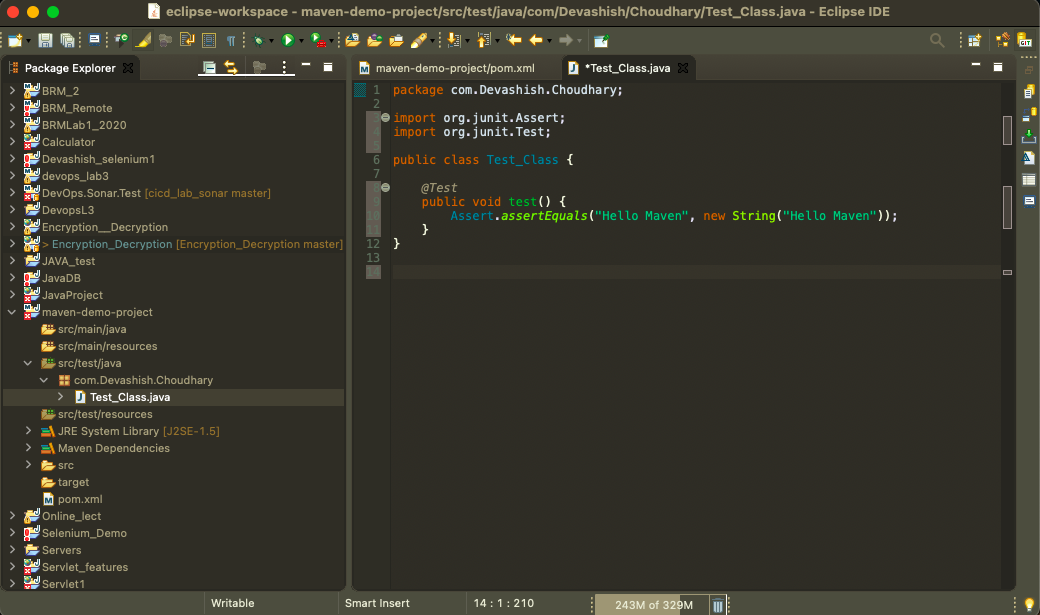
*@Test*

public void test() {

Assert.*assertEquals*("Hello Maven", new String("Hello Maven"));

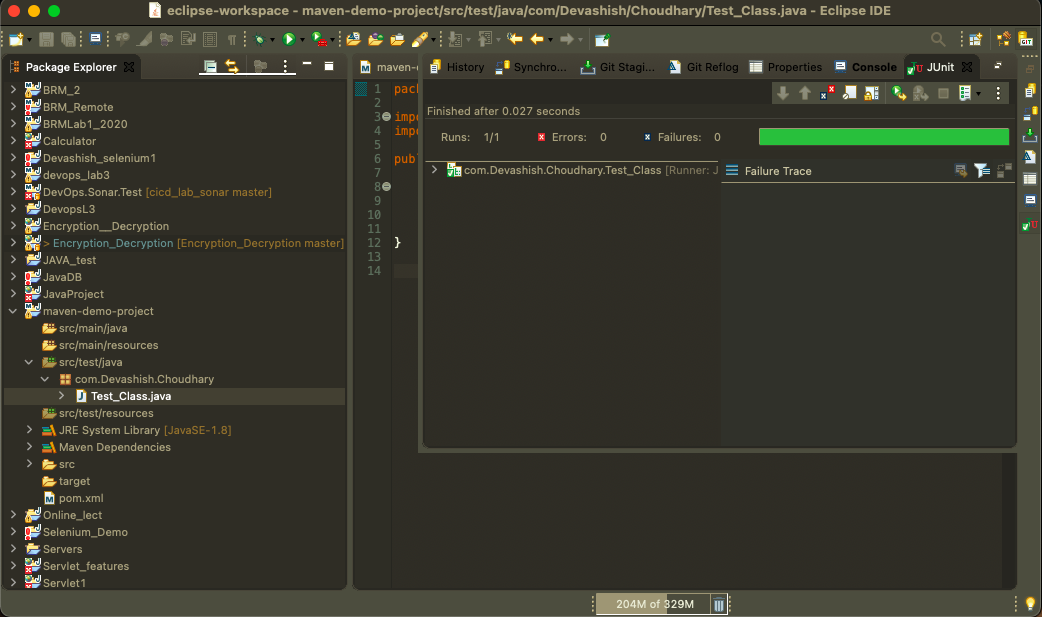
}

}



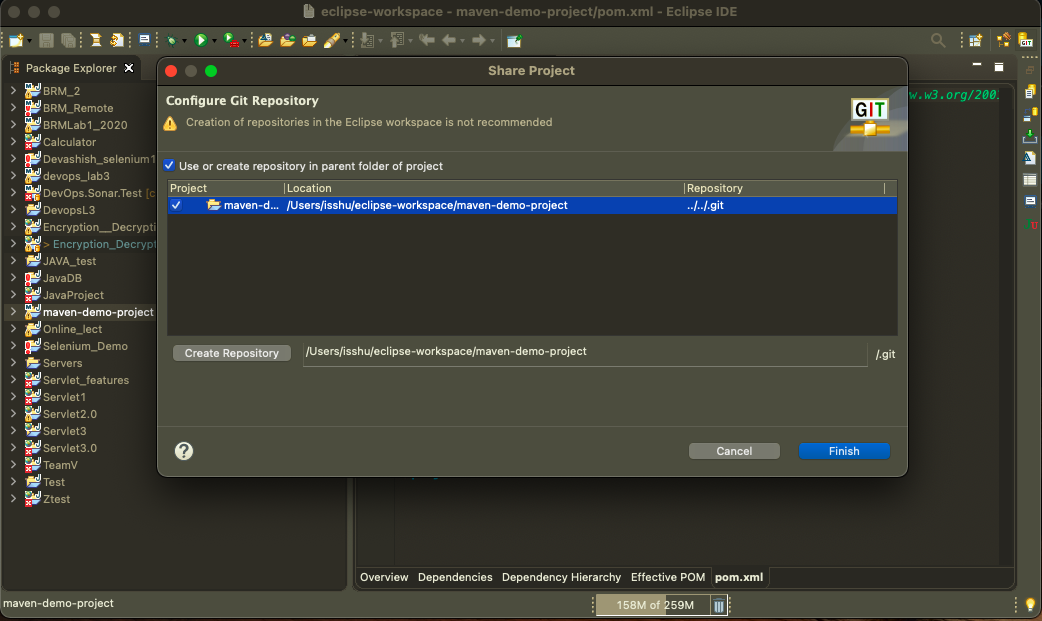
## Step-8

Run your first maven project. Right click on *AppTest.java* → Run as → JUnit Test.

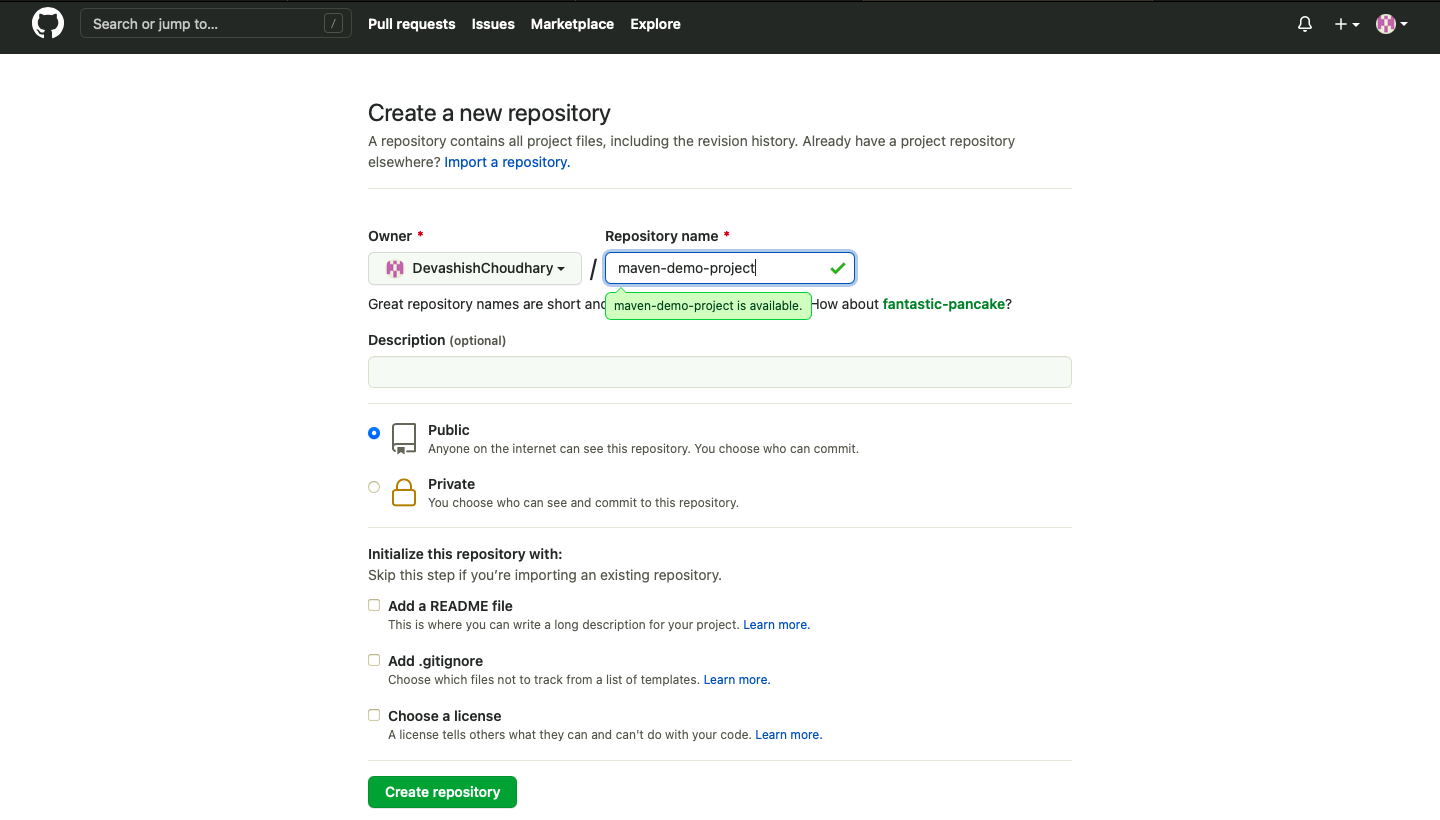


***Push the same running project on GitHub using Eclipse or IDE.***

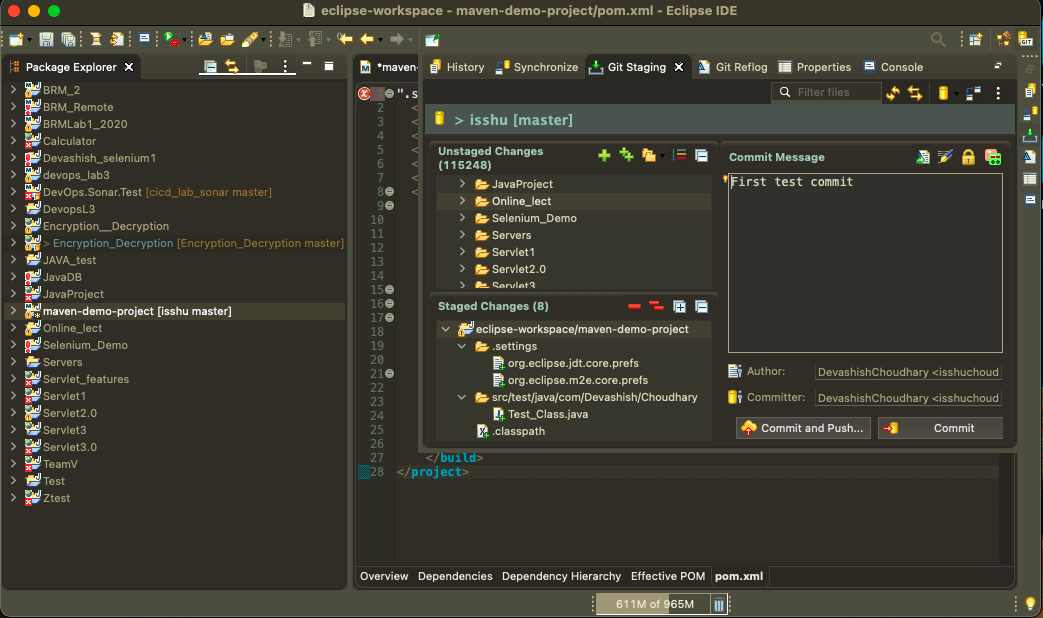
* Right click on the project and choose *Team* **->** *Share* **->** *Git*.

******

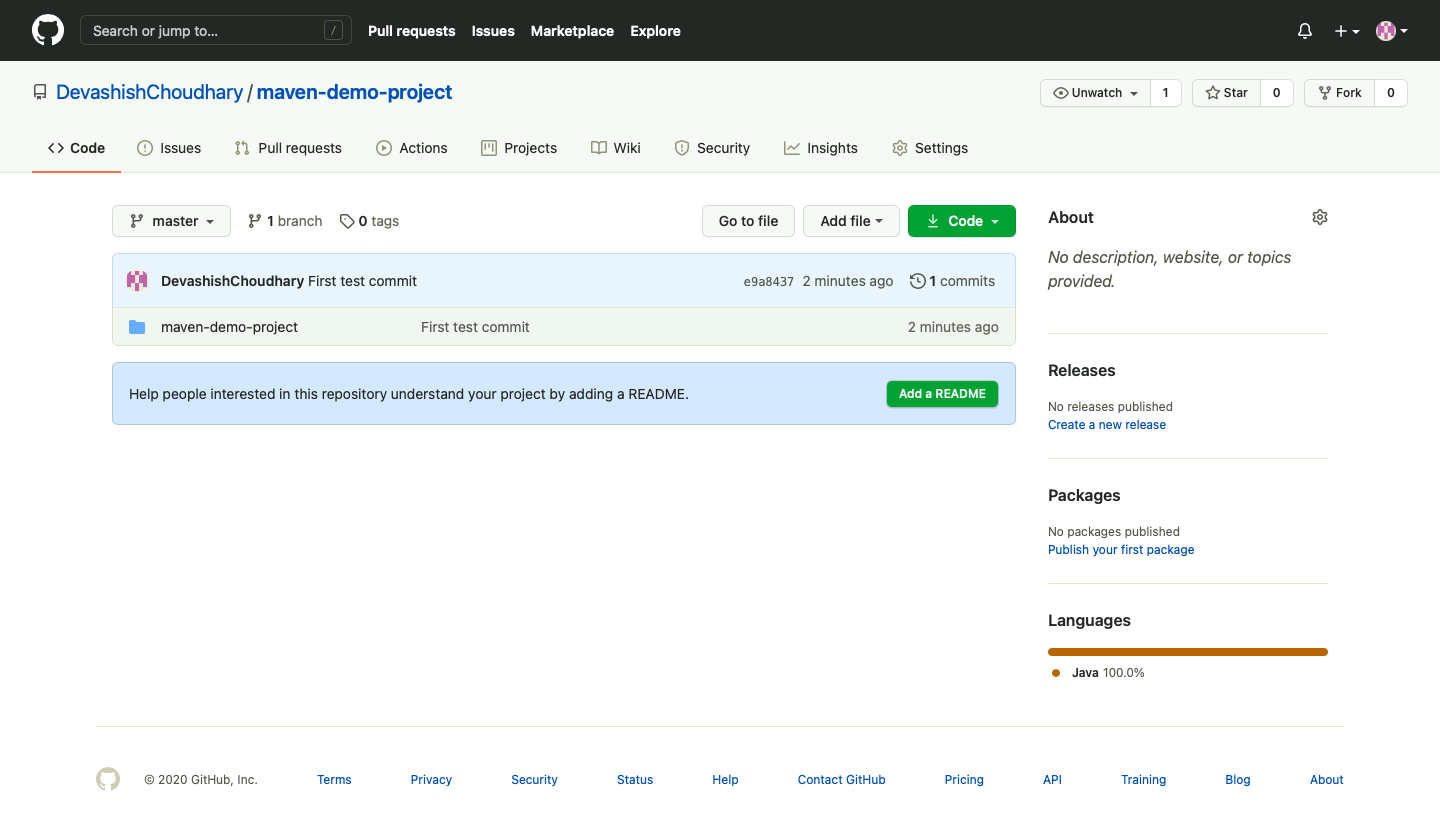
* **Create a new repository on your GitHub account with the same new of your project have.**



* Right click again to *Project* ->*Team* -> *commit* and move your project from **unstaged changes** to **staged changes** then **you are good to go for doing commit and push.**

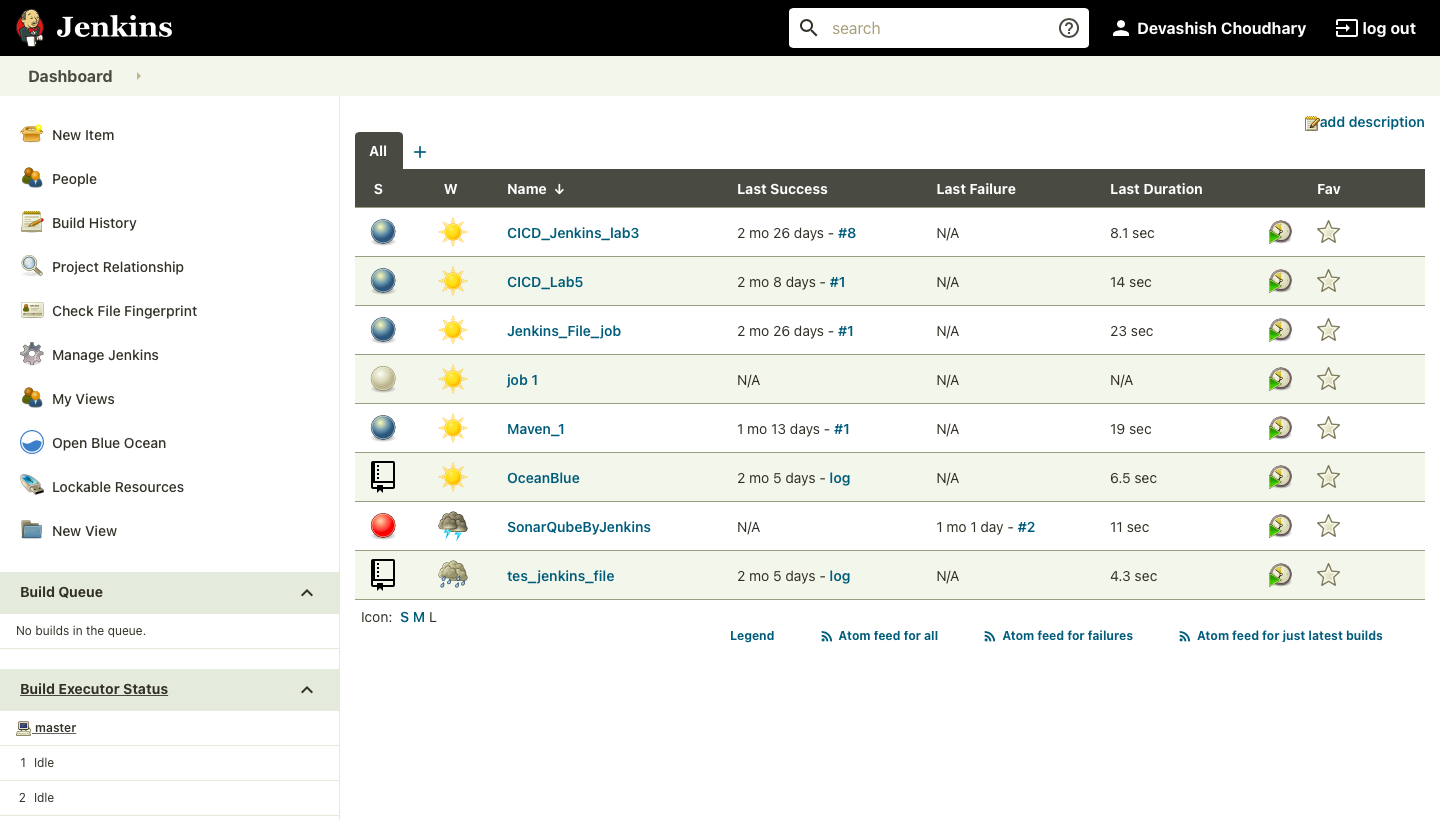


* Now you can see the project on your GitHub repository.

******

***Create a Jenkins Pipeline to generate build***

* Start the Jenkins service using the java -jar jenkins.war command in the command prompt and then go to the Jenkins dashboard at localhost:8080

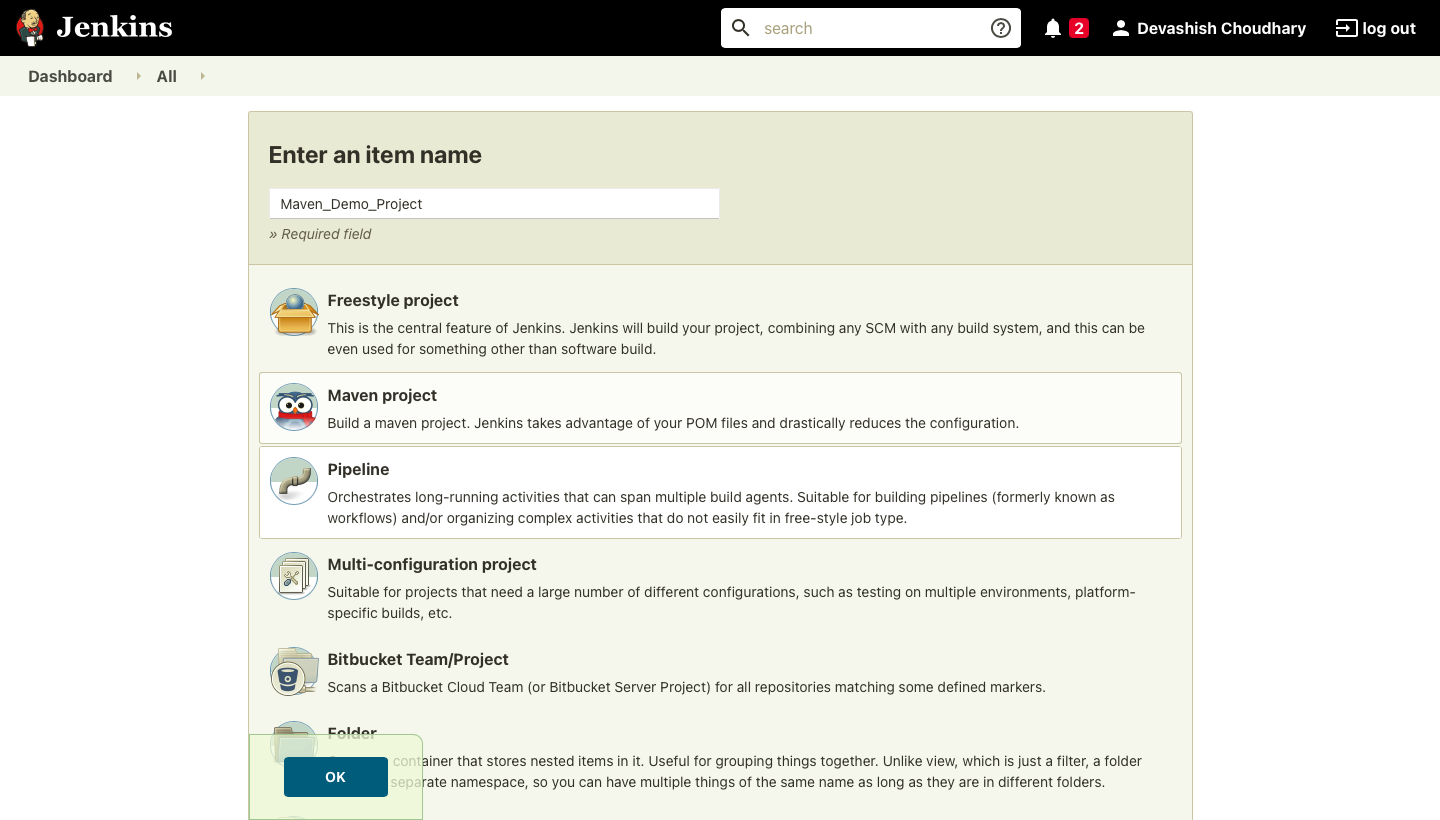
******

* Click on the new item option.

Select the pipeline option.

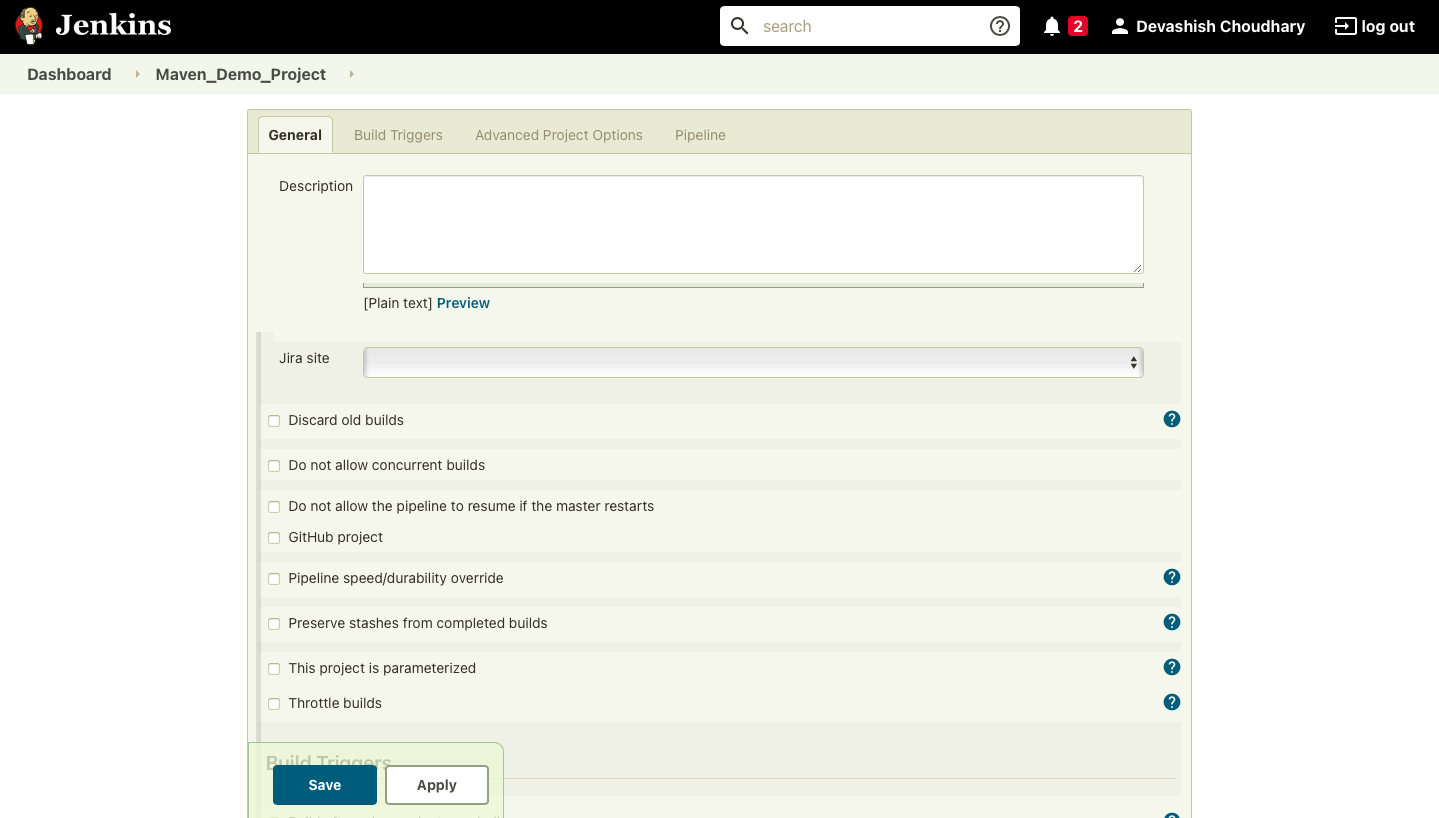
Give a name to the project (In my case the name is “Maven\_Demo\_Project”).

Click ok.



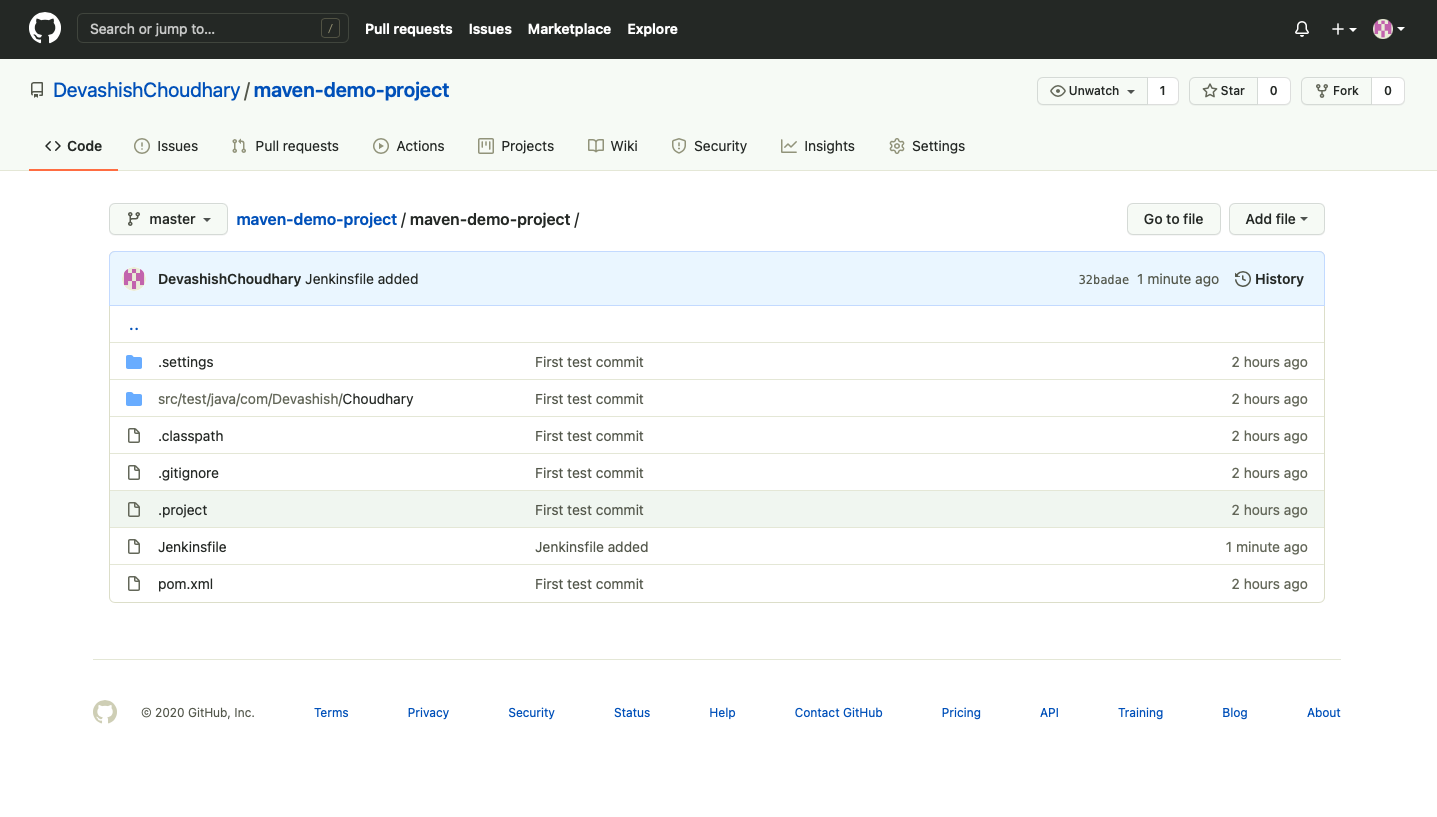
* You will be directed to a page similar shown below.

Click on the source code management tab to add the GitHub repository.

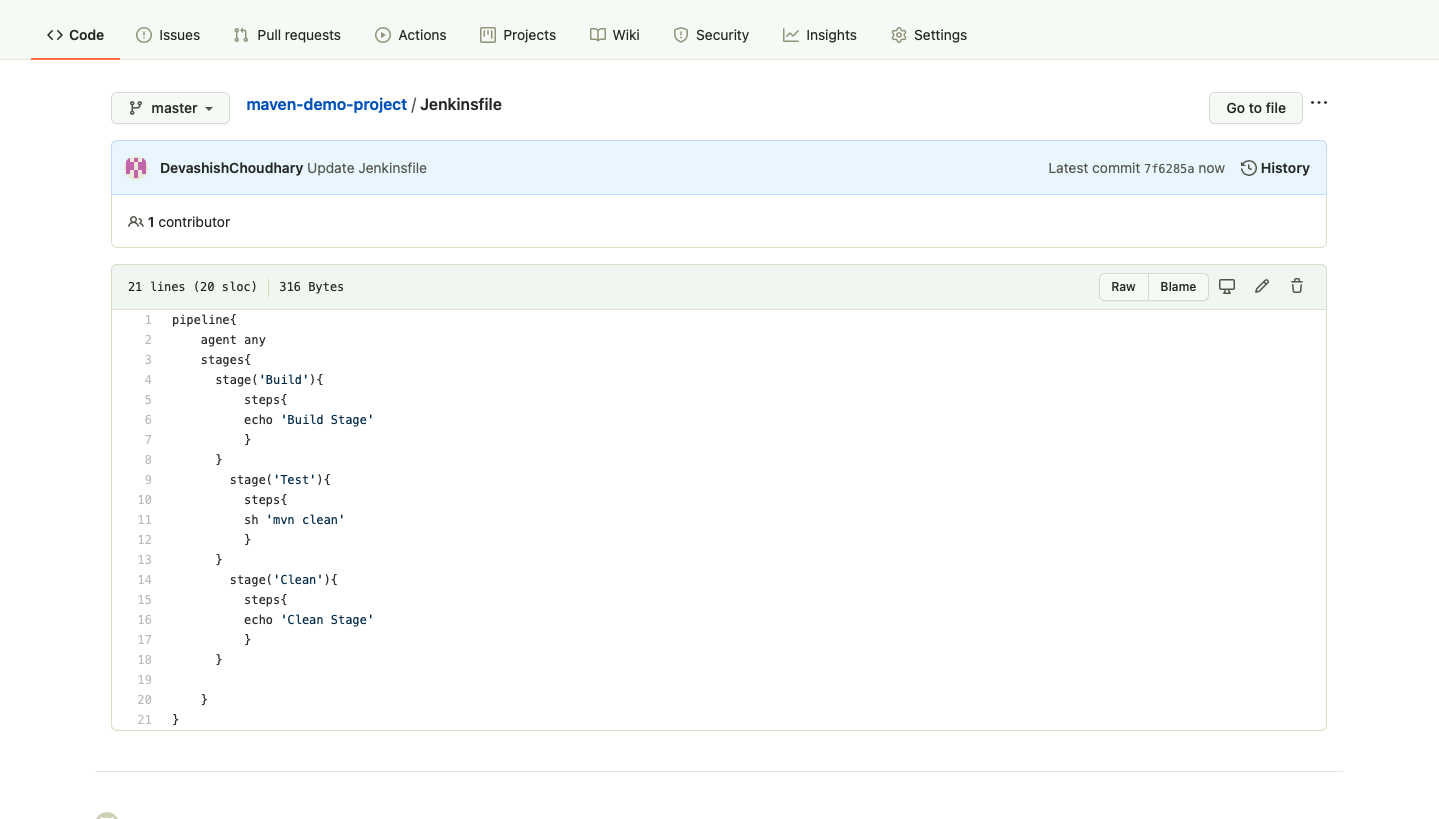


* Visit the repository where you saved your maven project

Copy the location of the GitHub repository



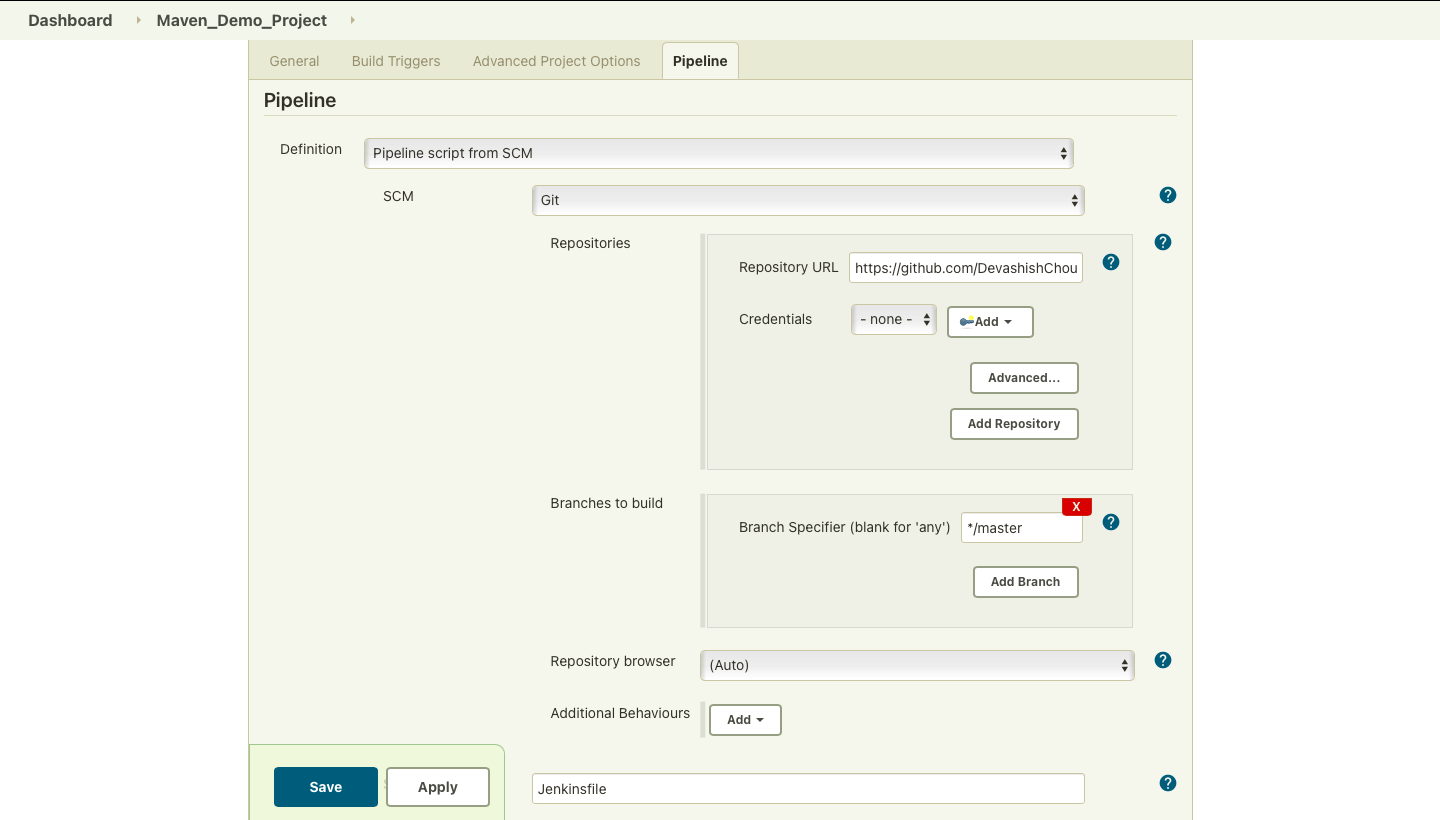
* The github repository has to contain a file named “Jenkinsfile” that tells the Jenkins server what tasks to do. In this example, it asks the Jenkins server to run 3 commands in 3 stages namely:
* echo “Build Stage”
* mvn clean
* echo “Clean Stage”



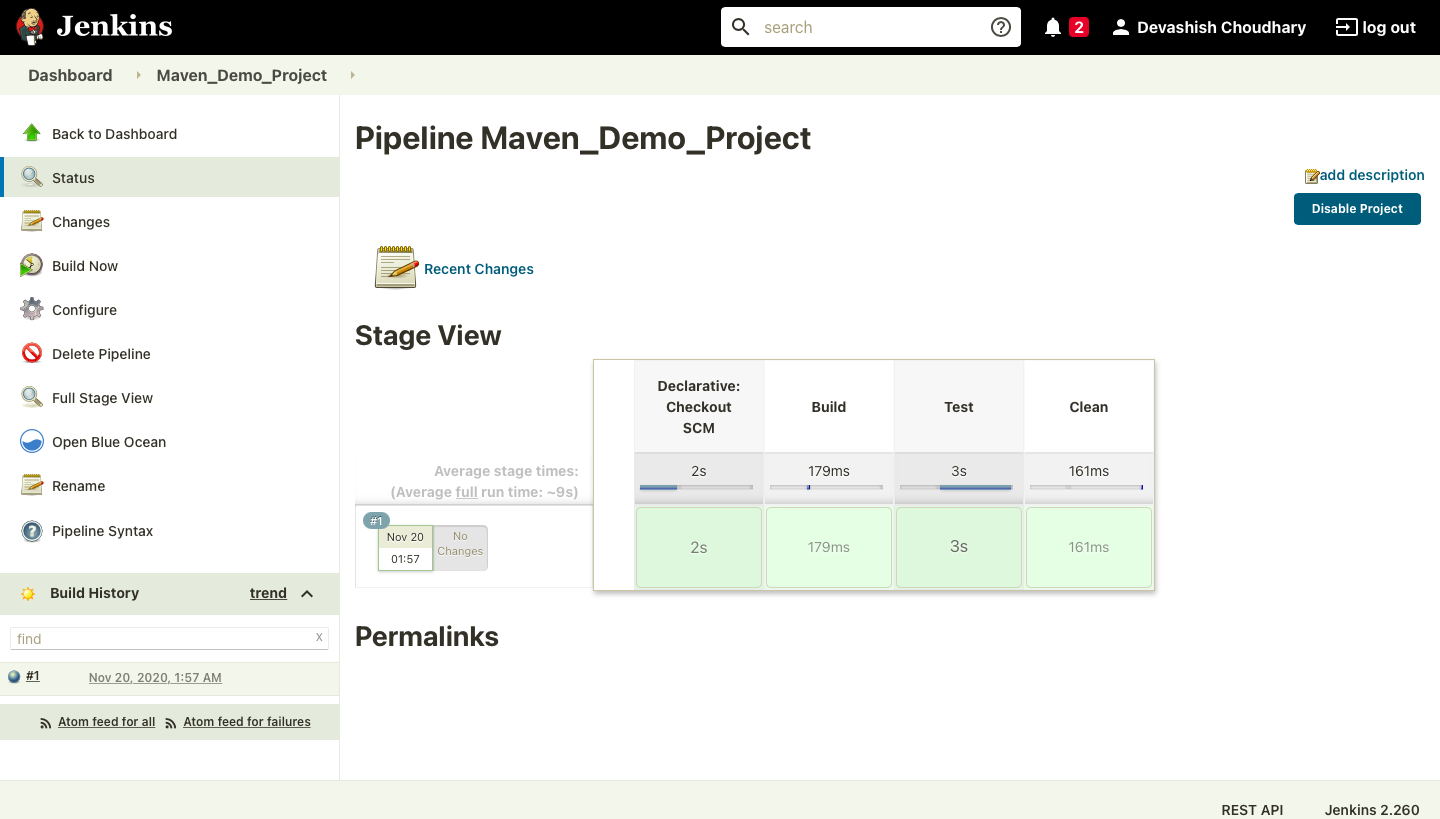
* In the Source Code Management Section select Git.

Give the repository location in the specified placeholder.

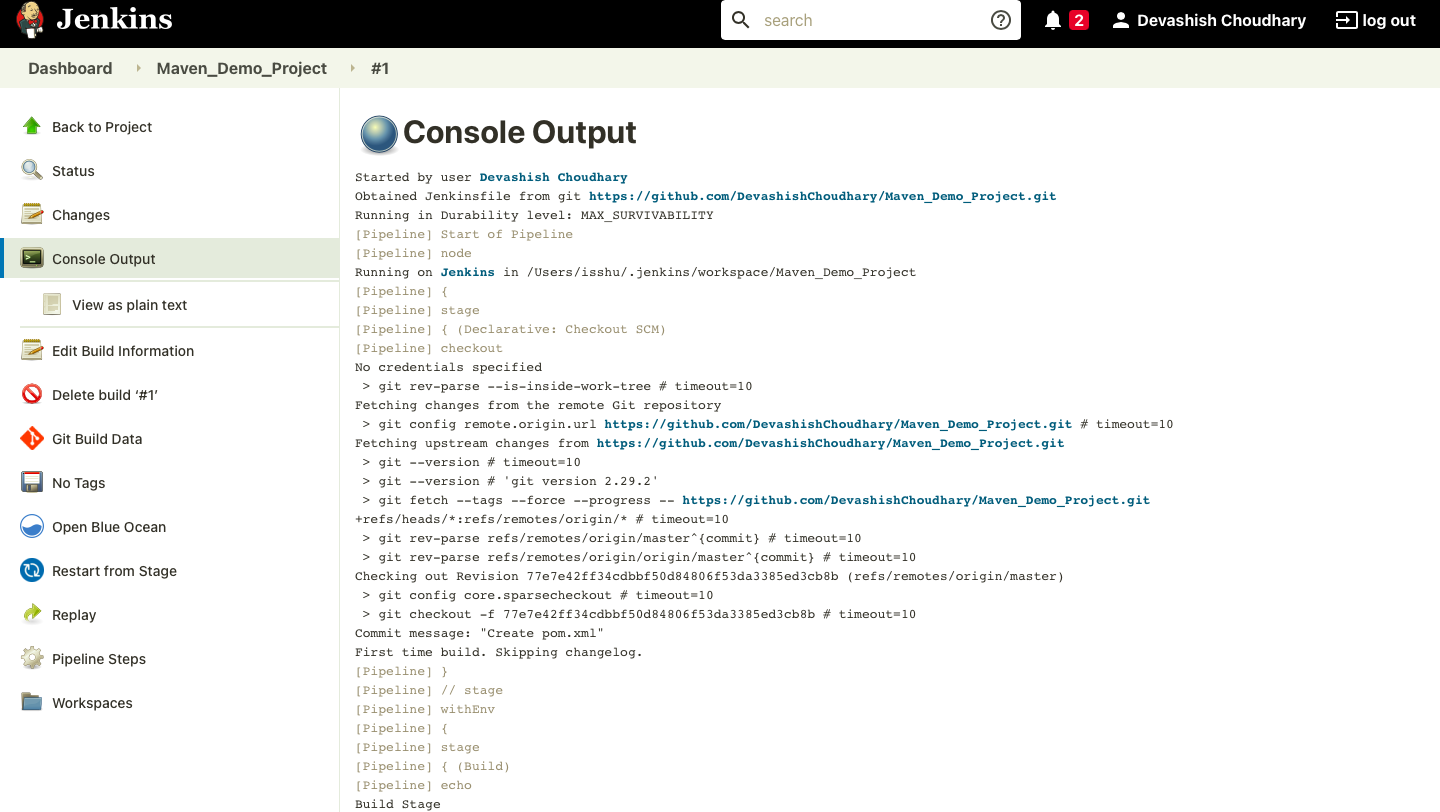
Click save and apply.



* You would be redirected to a dashboard like this after the build.



Console Output as shown below.

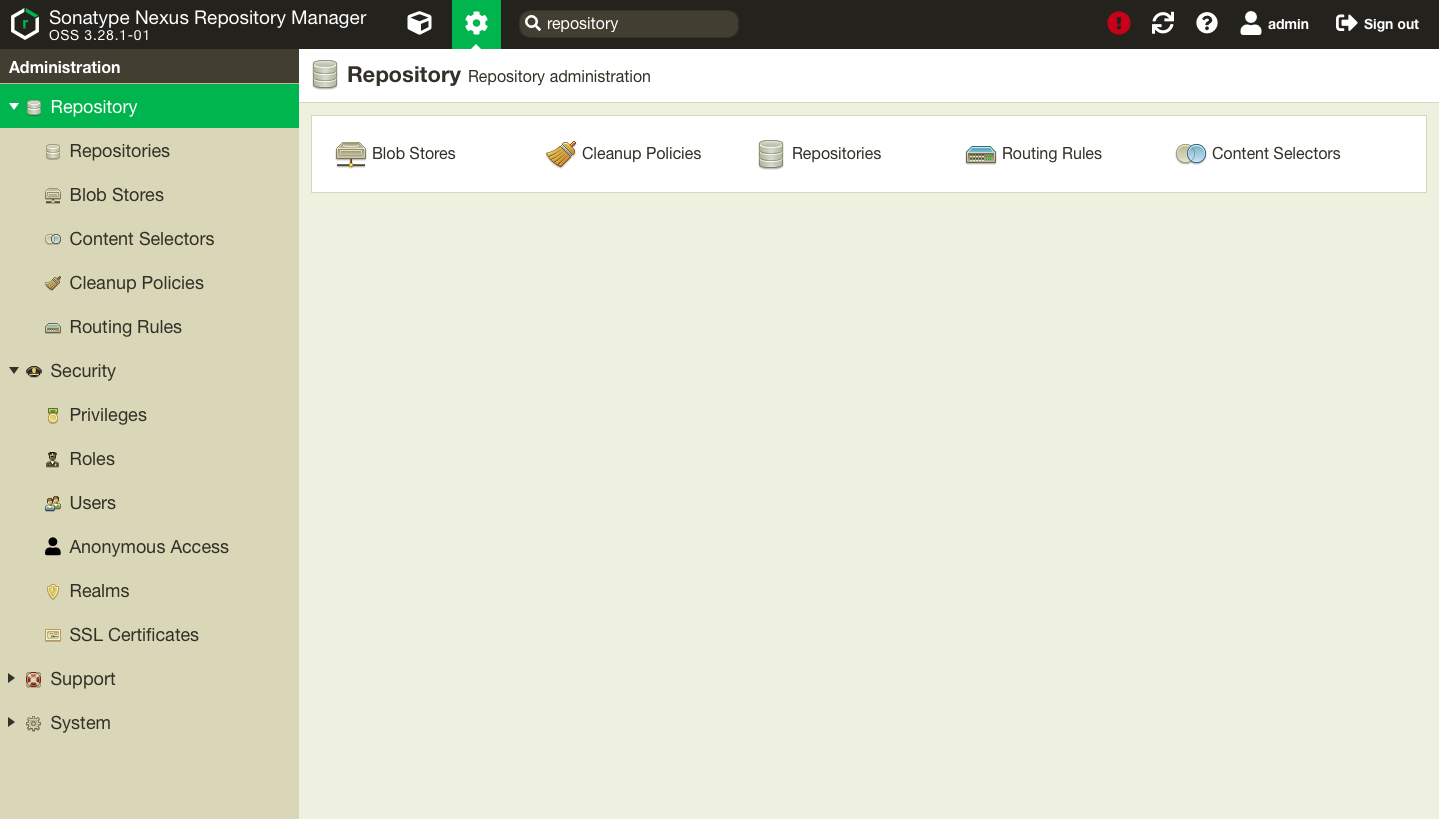


* If the build runs successfully it would return “BUILD SUCCESS”. It would return failure in case of any errors in the project.

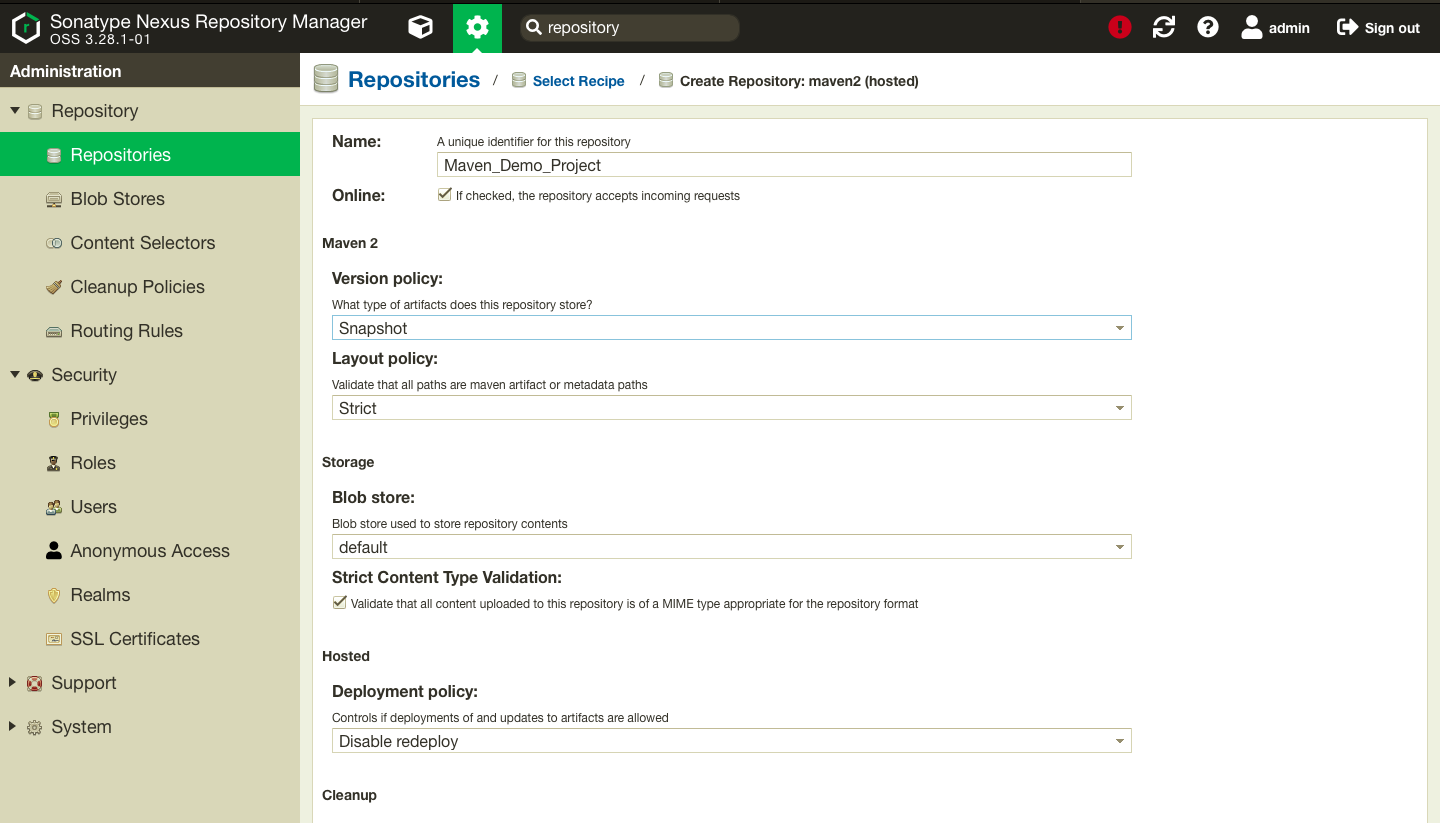


***Uploading Artifact on Nexus***

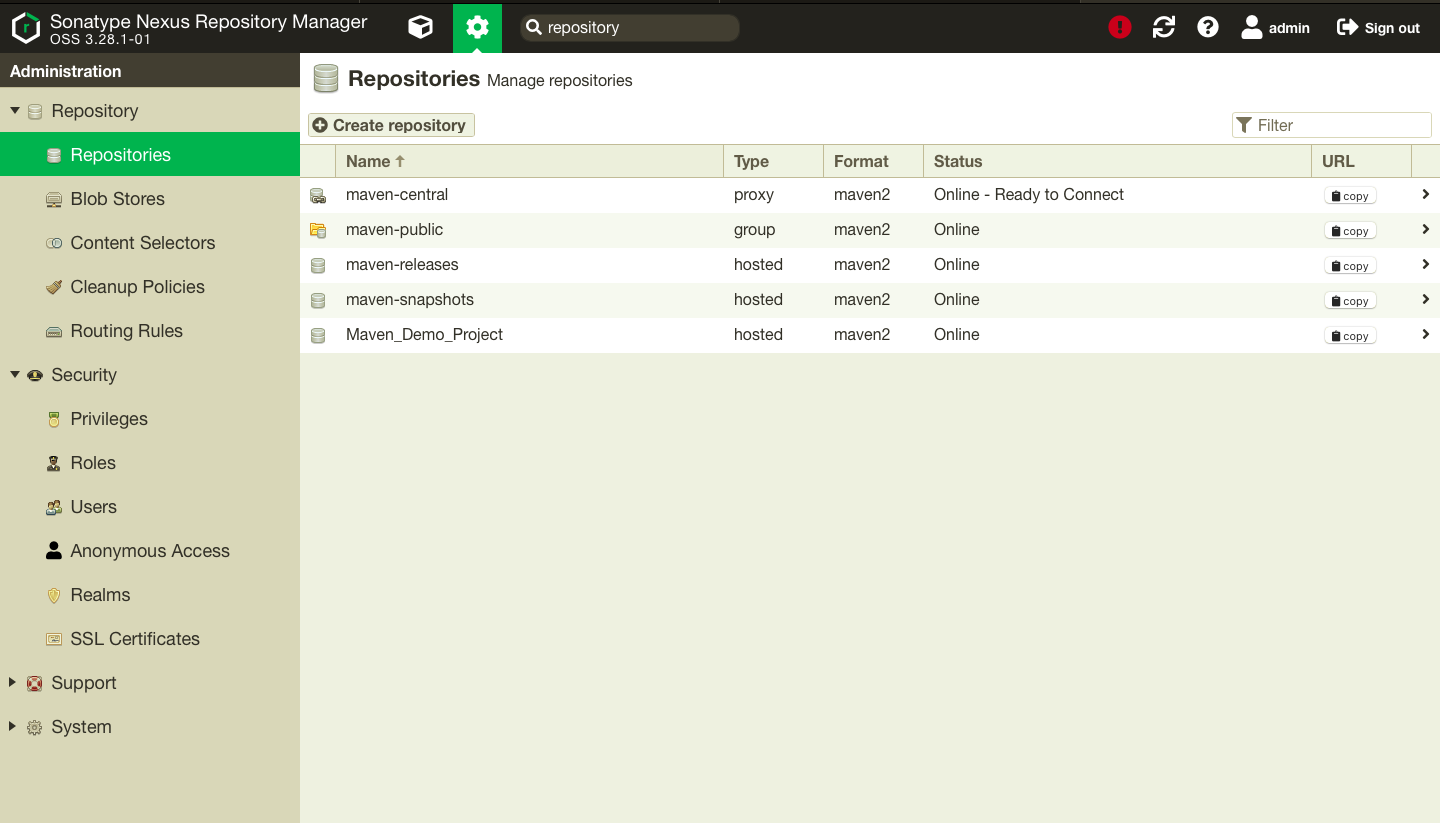
* Creating repository on Nexus as mention below:



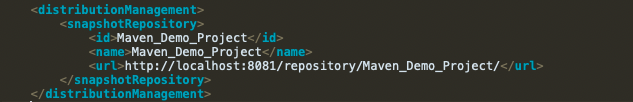
* Configure the new repository using “*maven2 (hosted)”*as shown below:



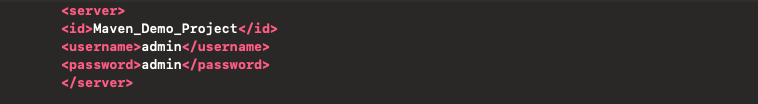
* Now you can see the created repository on your nexus Dashboard.



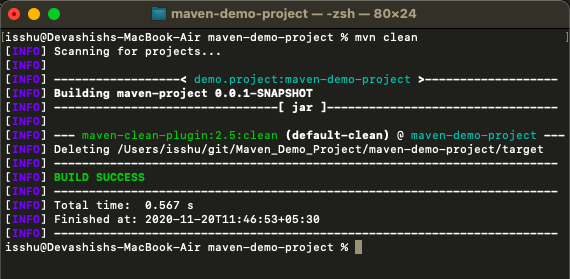
* Add the following code in the pom.xml file of your project accordingly to link the project with nexus.



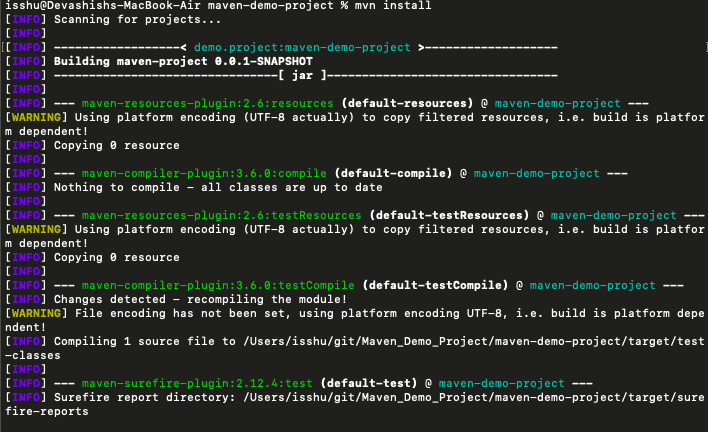
* Add the following server code in the *apache-maven-3.6.3/conf/setting.xml*



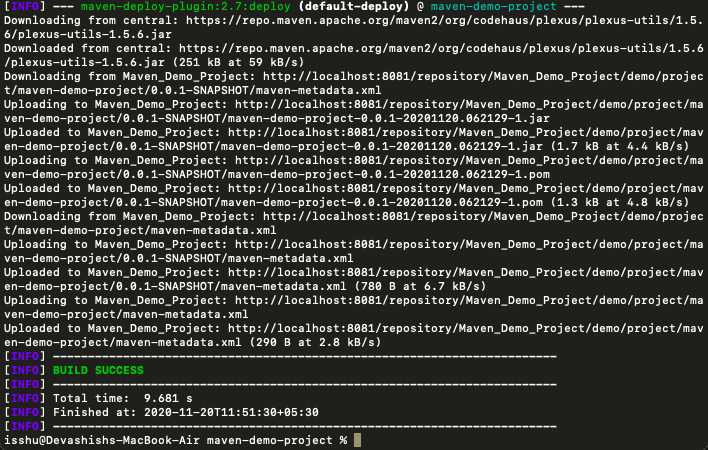
* See the following console output with different command:
* mvn clean



* mvn install



* mvn deploy



* Finally you have see all the report of your project on the dashboard of the nexus.

