



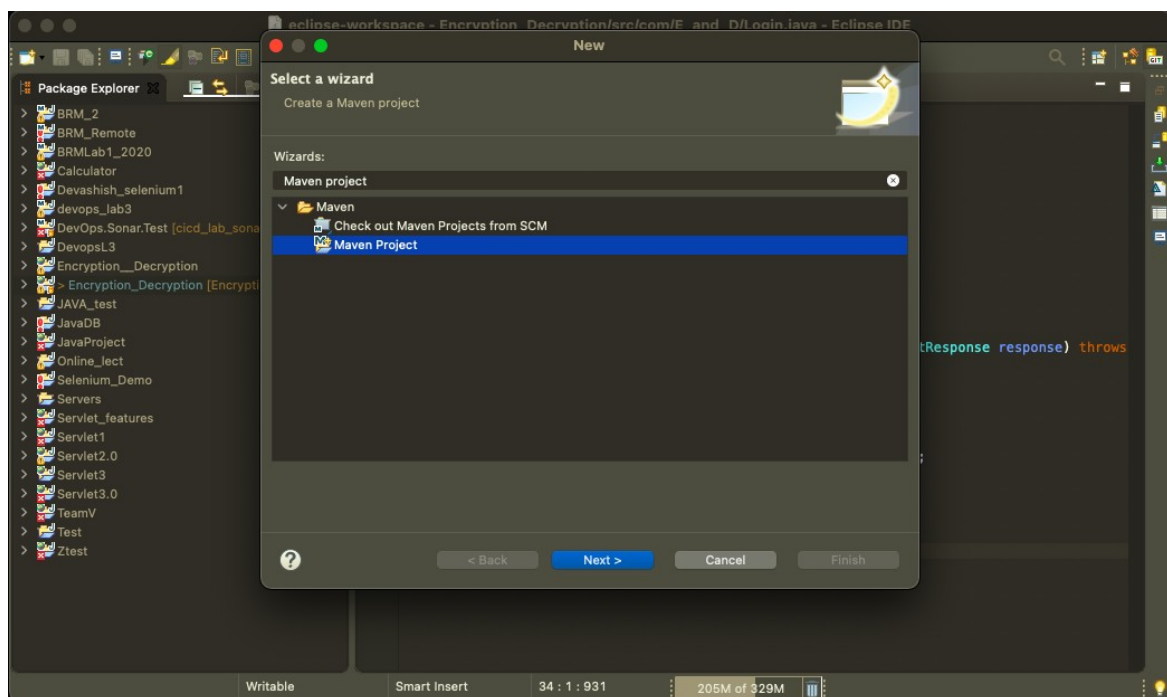
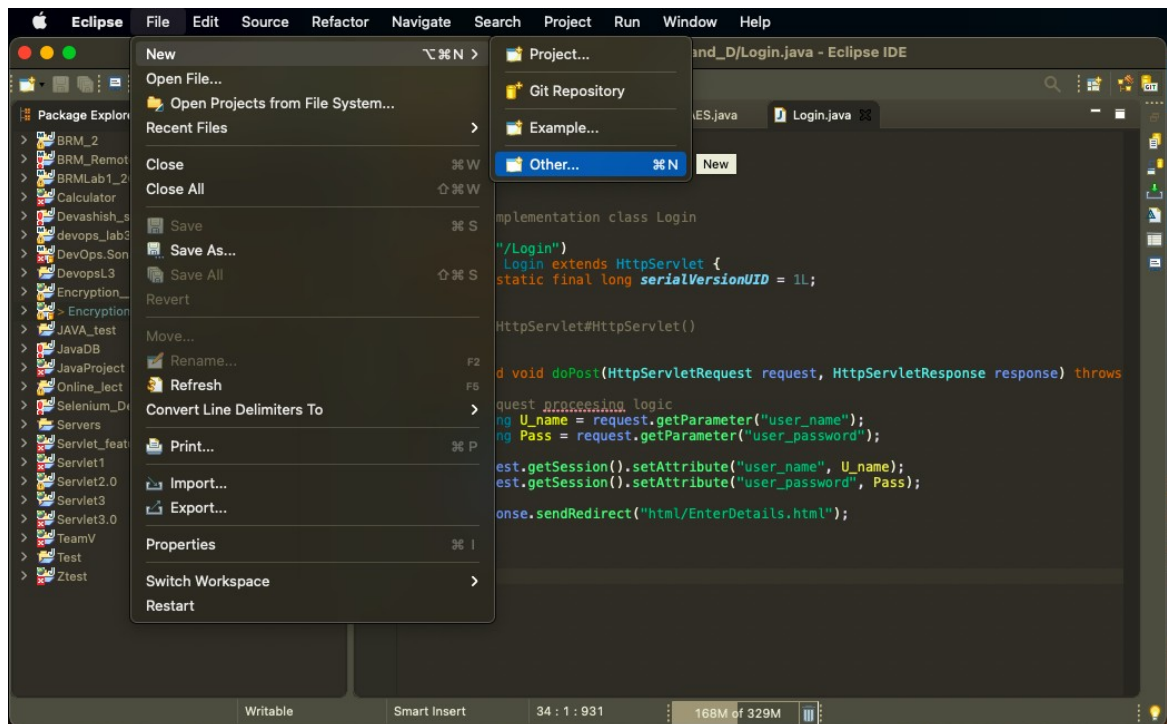
**Name - Devashish Choudhary**  
**Roll.No – R171218122**  
**SAP ID – 500070510**  
**DevOps Batch- 2 (5<sup>th</sup> Semester)**  
**Submitted to- Mr. Hitesh Kumar**

**CICD – Assignment 2**

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

### Step-1

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

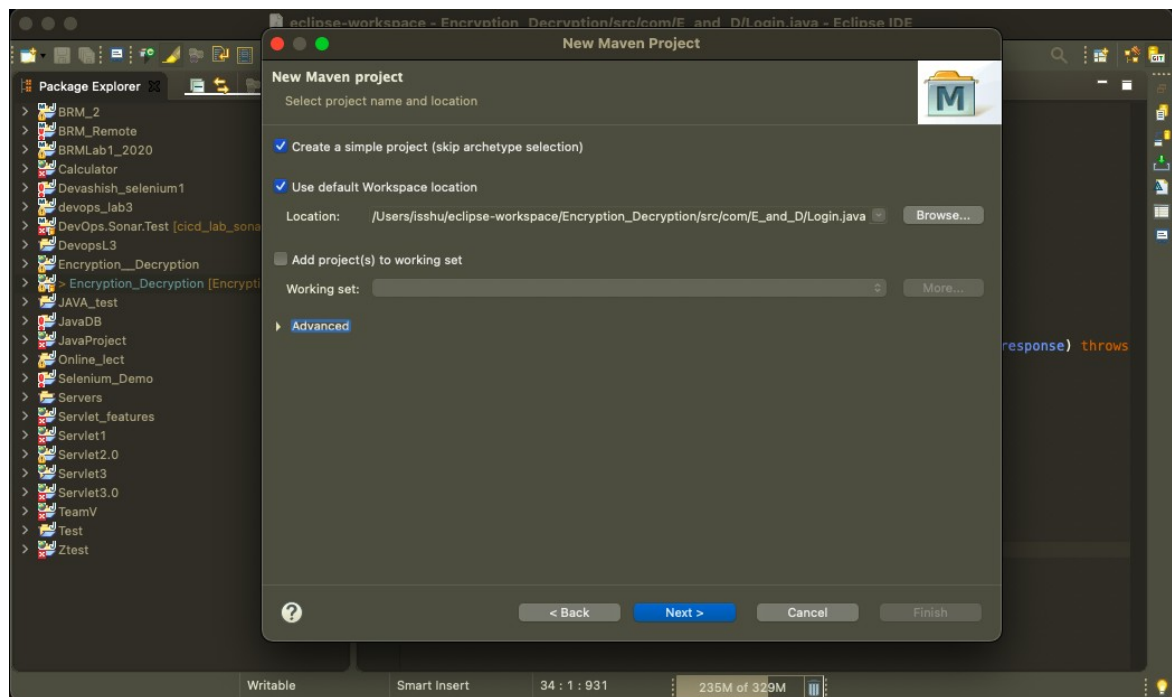


## Step-2

Click on Checkbox for both

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

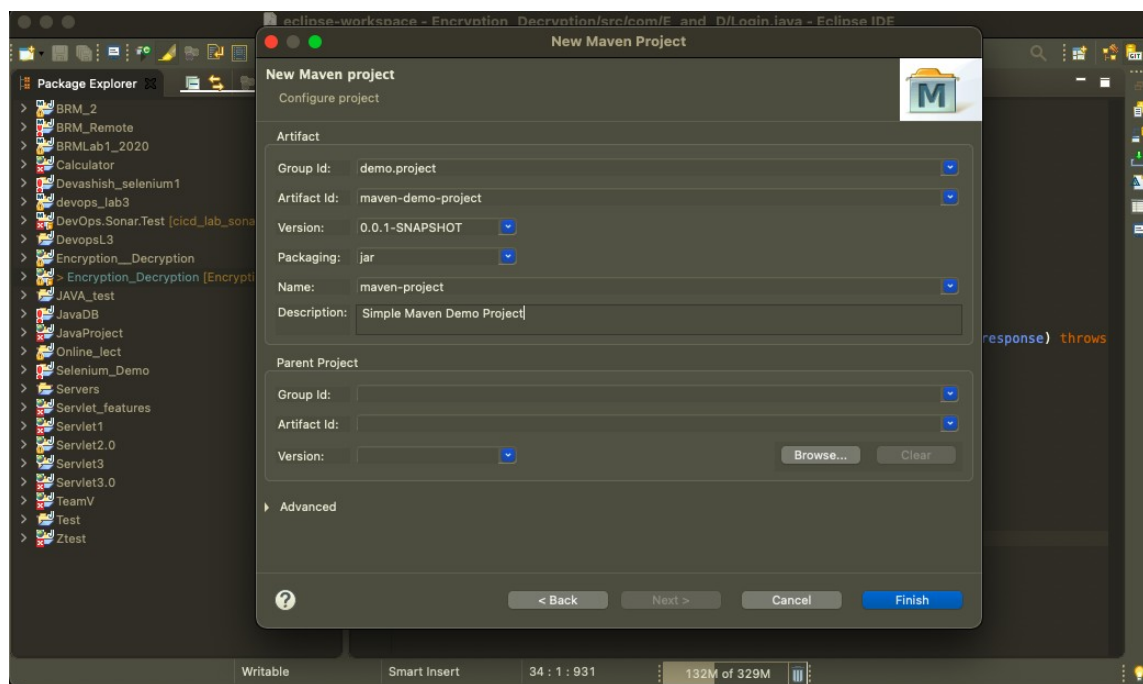
Note:- Check the option “Create a simple project (skip archetype selection)”.



### Step-3

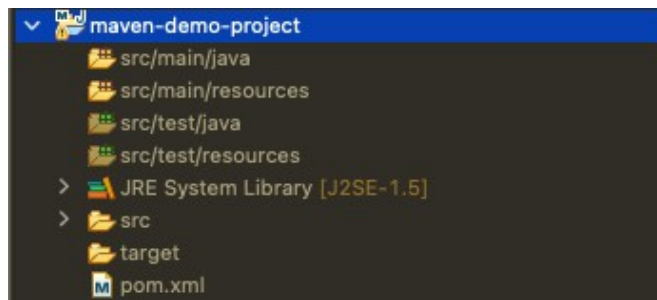
Provide GroupId and ArtifactId in next screen.

- **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

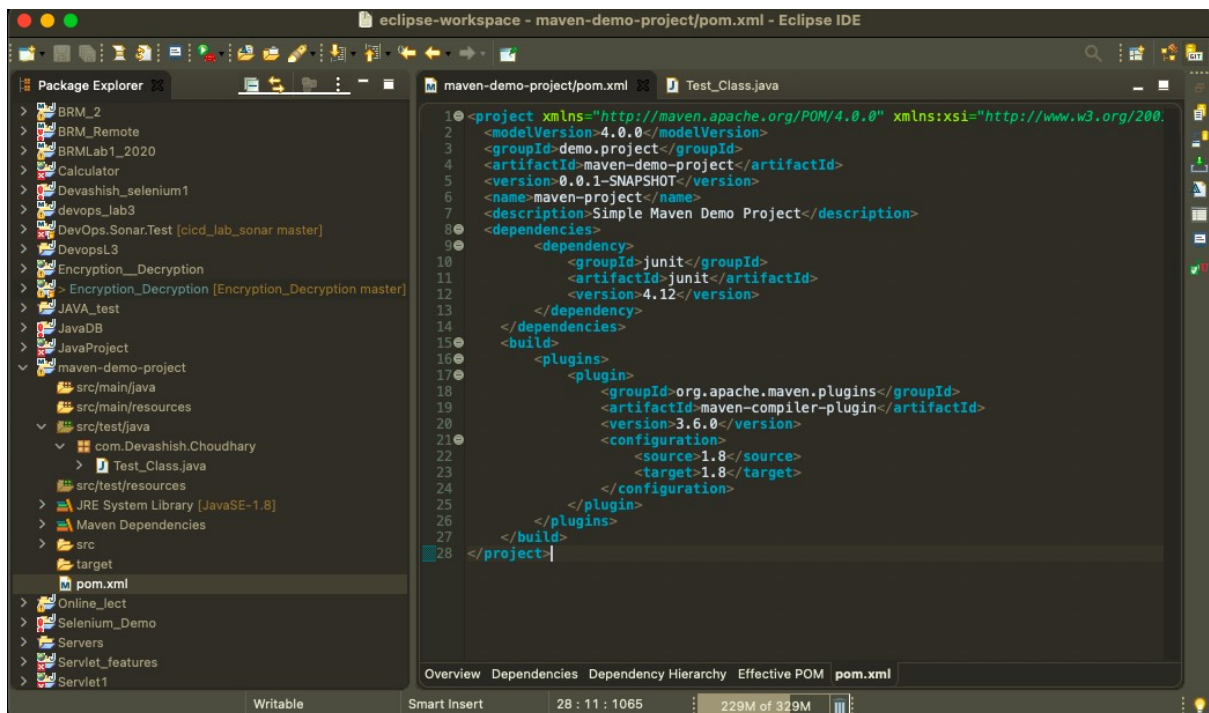
Add some dependencies to **pom.xml** file. Here we are adding *junit* dependency to **pom.xml**.

```
<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.

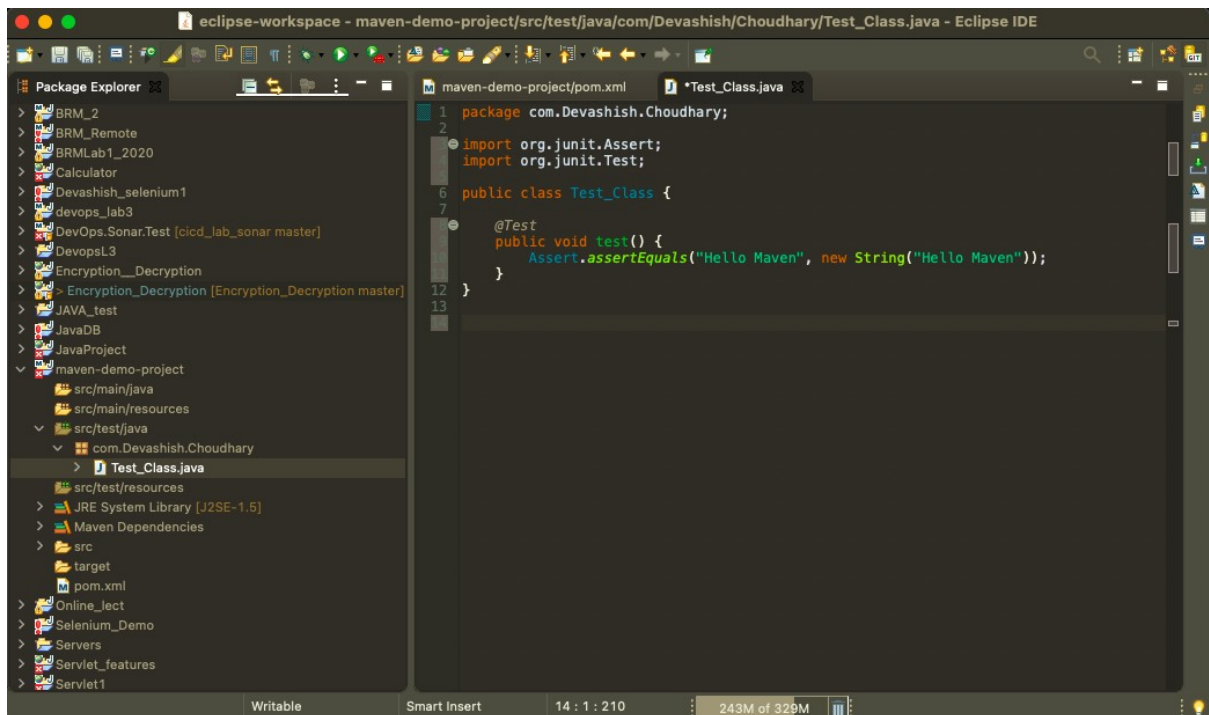
```
package com.Devashish.Choudhary;

import org.junit.Assert;
import org.junit.Test;

public class Test_Class {

    @Test
    public void test() {
        Assert.assertEquals("Hello Maven", new String("Hello Maven"));
    }

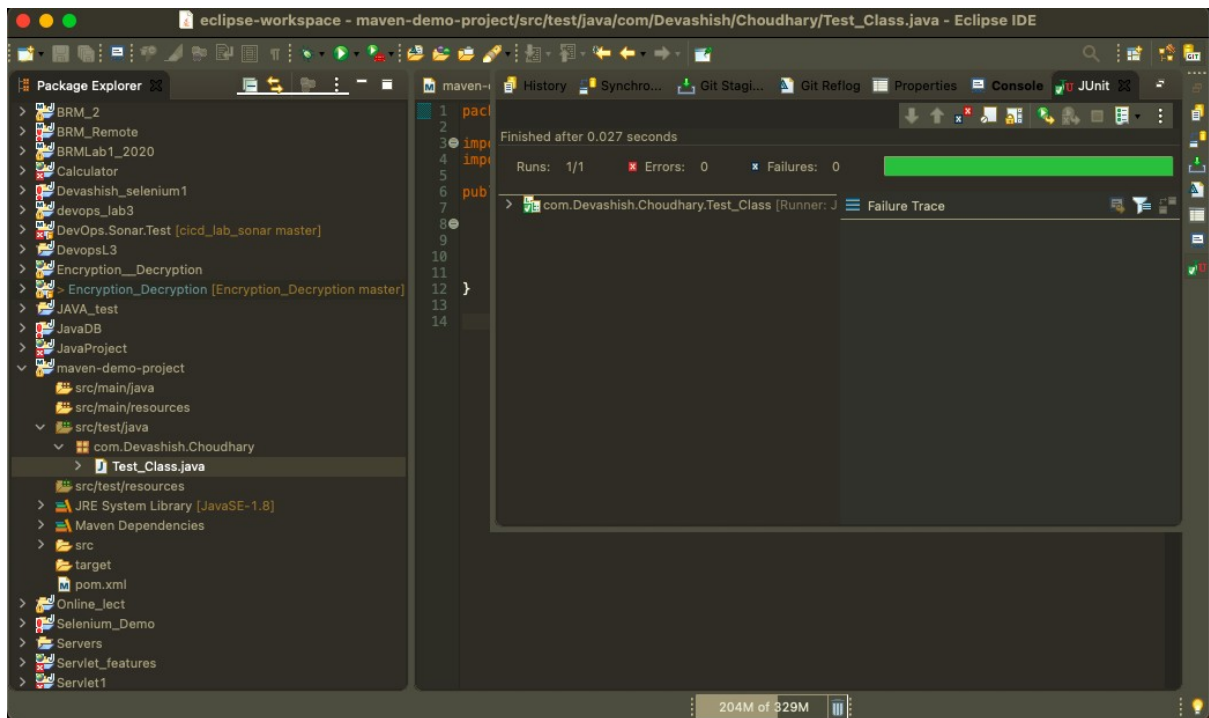
}
```



## Step-8

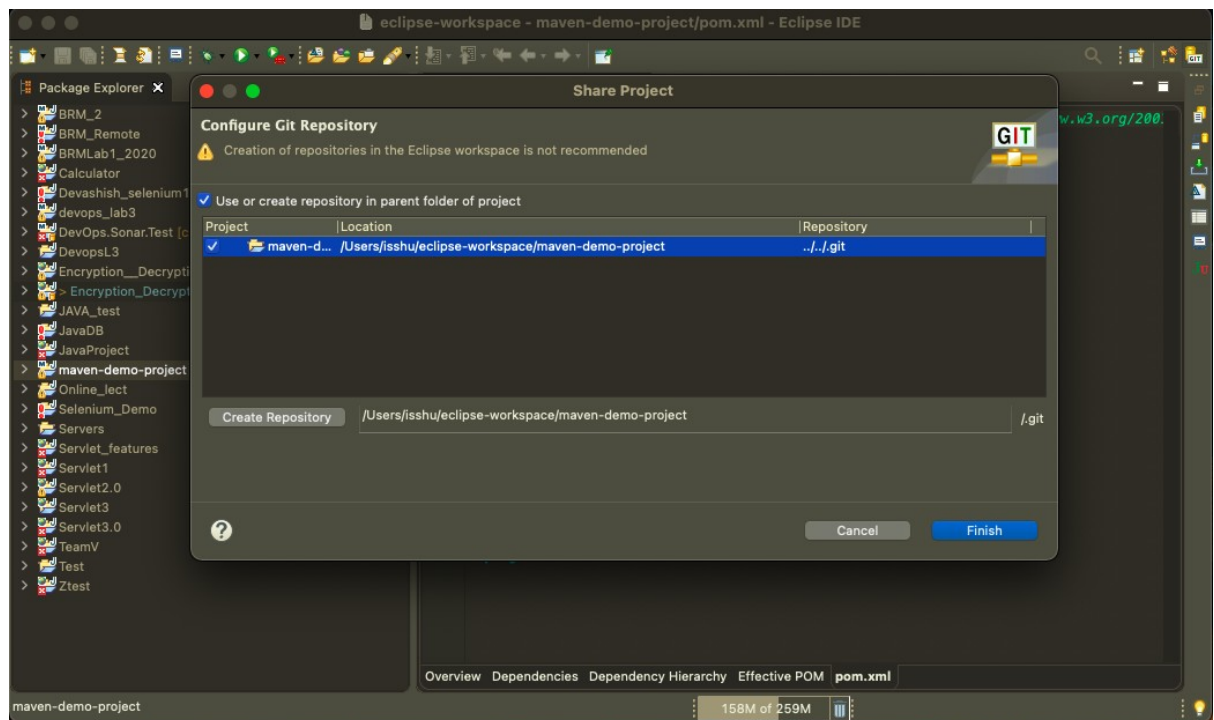
Run your first maven project. Right click on *AppTest.java* → Run as → JUnit Test.





### **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.



Search or jump to... Pull requests Issues Marketplace Explore

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \* Repository name \*

DevashishChoudhary / maven-demo-project ✓

Great repository names are short and [maven-demo-project is available](#). How about [fantastic-pancake?](#)

Description (optional)

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

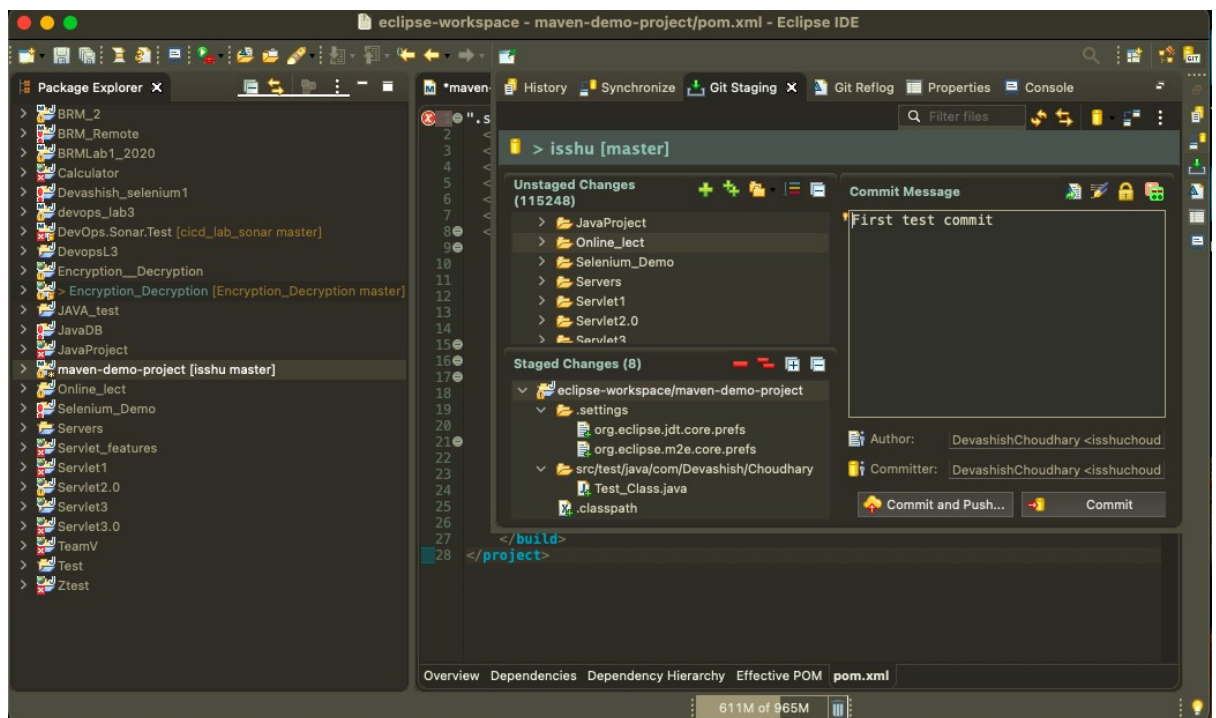
☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

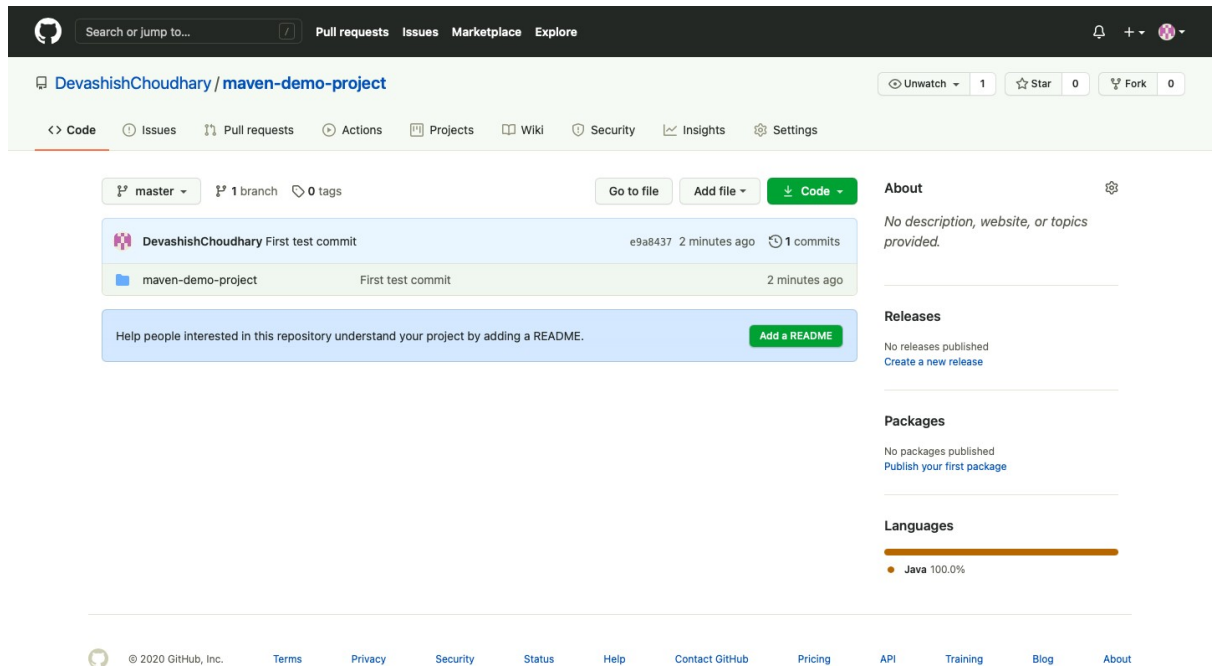
☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

[Create repository](#)

- 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.

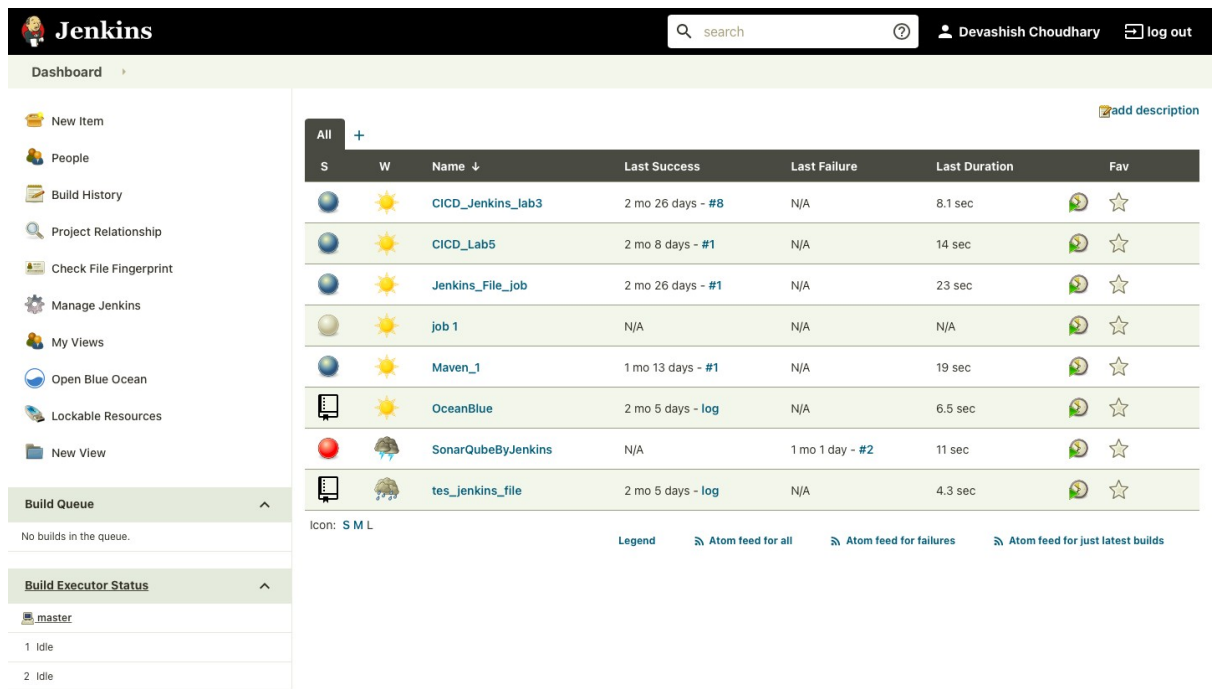


- Now you can see the project on your GitHub repository.



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

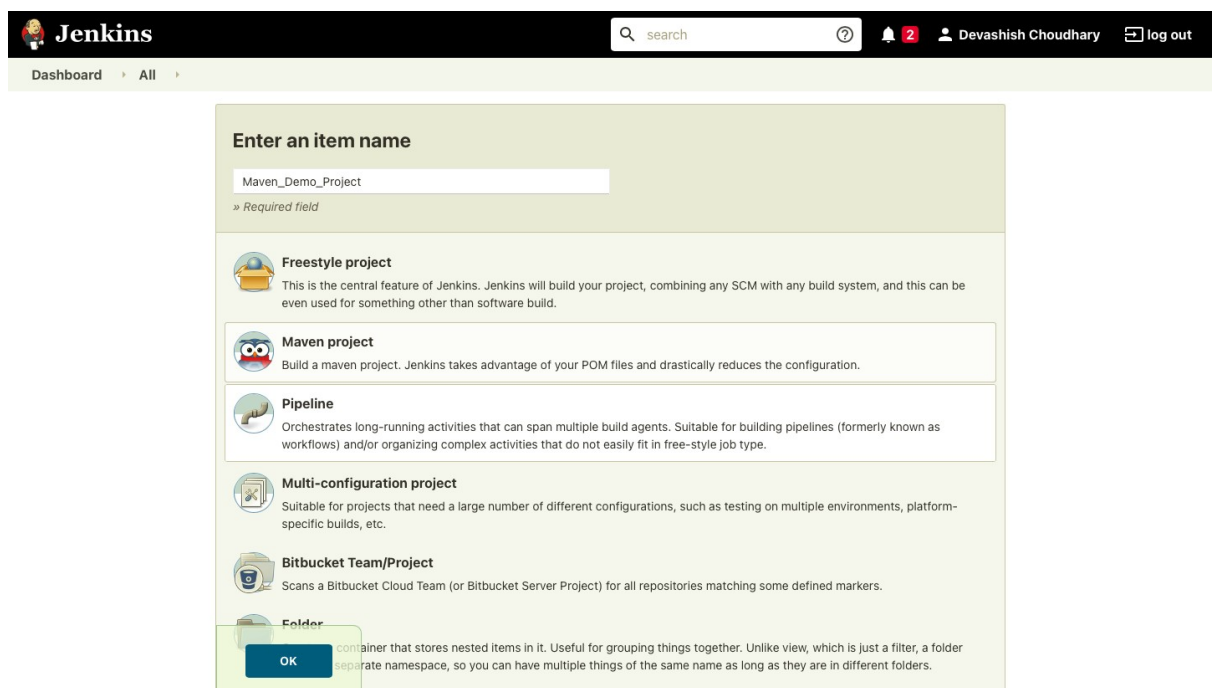
- Start the Jenkins service using the `java -jar jenkins.war` command in the command prompt and then go to the Jenkins dashboard at `localhost:8080`



The screenshot shows the Jenkins Dashboard. On the left is a sidebar with navigation options: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Open Blue Ocean, Lockable Resources, and New View. The main area displays a table of jobs. Below the table are sections for 'Build Queue' (showing no builds) and 'Build Executor Status' (showing two idle executors).

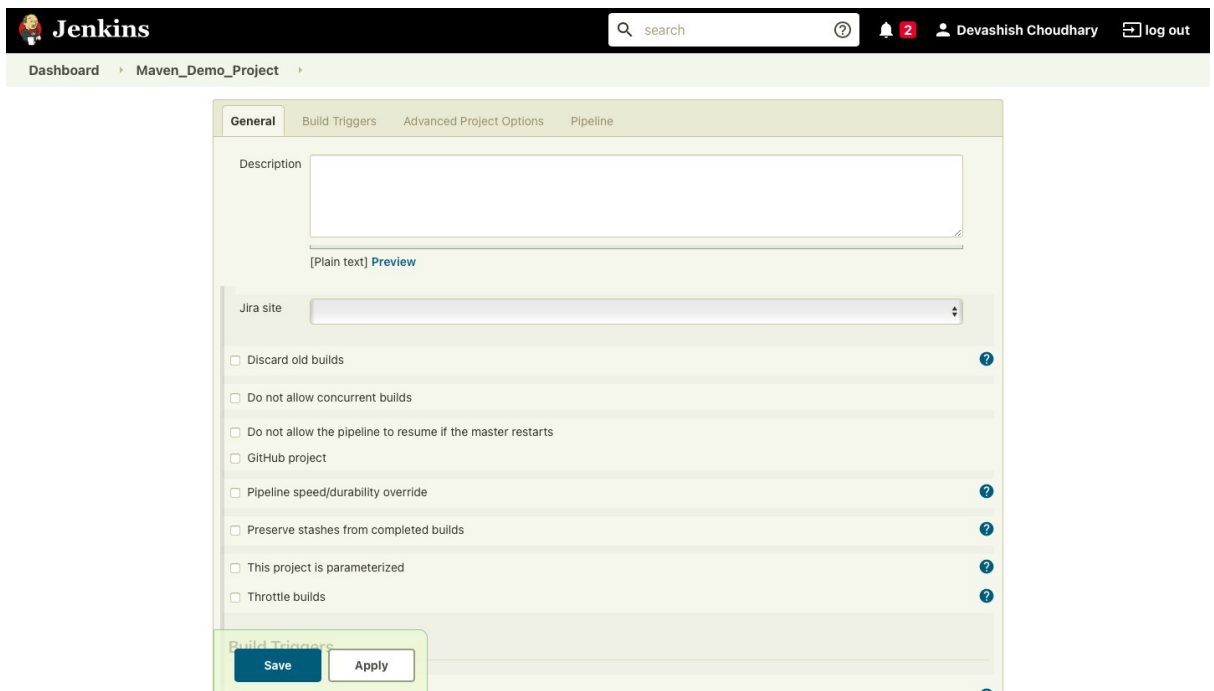
| S | W | Name ↓             | Last Success      | Last Failure    | Last Duration | Fav |
|---|---|--------------------|-------------------|-----------------|---------------|-----|
|   |   | CICD_Jenkins_lab3  | 2 mo 26 days - #8 | N/A             | 8.1 sec       |     |
|   |   | CICD_Lab5          | 2 mo 8 days - #1  | N/A             | 14 sec        |     |
|   |   | Jenkins_File_job   | 2 mo 26 days - #1 | N/A             | 23 sec        |     |
|   |   | job 1              | N/A               | N/A             | N/A           |     |
|   |   | Maven_1            | 1 mo 13 days - #1 | N/A             | 19 sec        |     |
|   |   | OceanBlue          | 2 mo 5 days - log | N/A             | 6.5 sec       |     |
|   |   | SonarQubeByJenkins | N/A               | 1 mo 1 day - #2 | 11 sec        |     |
|   |   | tes_jenkins_file   | 2 mo 5 days - log | N/A             | 4.3 sec       |     |

- 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.

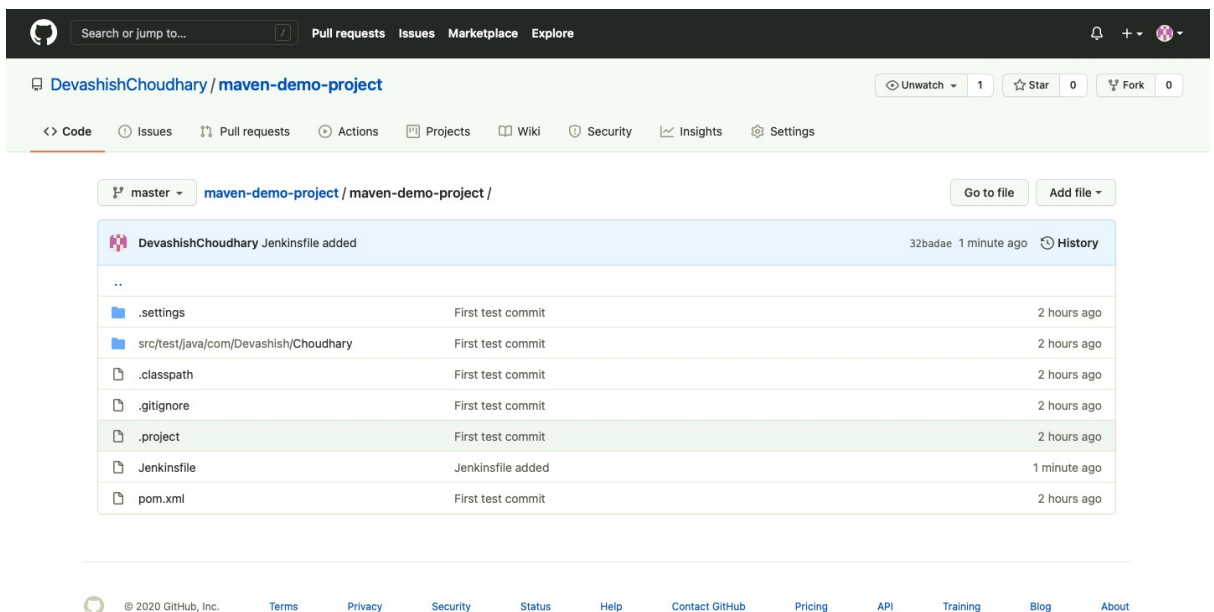


The screenshot shows the 'Enter an item name' dialog in Jenkins. The name 'Maven\_Demo\_Project' is entered in the text field. Below the field are several project type options: Freestyle project, Maven project, Pipeline, Multi-configuration project, Bitbucket Team/Project, and Folder. The 'Pipeline' option is highlighted with a green box. At the bottom, there is an 'OK' button.

- 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”

The screenshot shows the GitHub interface for the 'maven-demo-project / Jenkinsfile' repository. The file is 21 lines long (20 sloc) and 316 Bytes. The code defines a Jenkins pipeline with three stages: 'Build', 'Test', and 'Clean'.

```

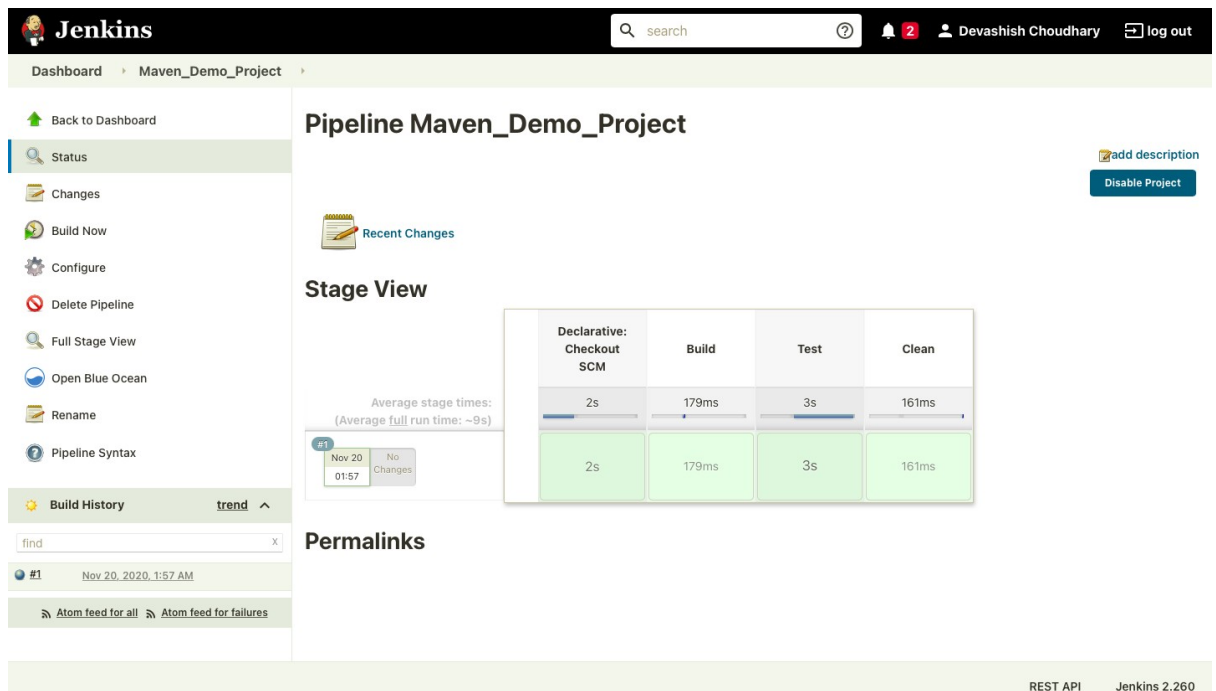
1 pipeline{
2   agent any
3   stages{
4     stage('Build'){
5       steps{
6         echo 'Build Stage'
7       }
8     }
9     stage('Test'){
10      steps{
11        sh 'mvn clean'
12      }
13    }
14    stage('Clean'){
15      steps{
16        echo 'Clean Stage'
17      }
18    }
19  }
20 }
21 }

```

- In the Source Code Management Section select Git. Give the repository location in the specified placeholder. Click save and apply.

The screenshot shows the Jenkins Pipeline configuration page for the 'Maven\_Demo\_Project'. The 'Definition' is set to 'Pipeline script from SCM'. The 'SCM' is set to 'Git'. The 'Repository URL' is 'https://github.com/DevashishChou'. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' set to '\*/master'. The 'Repository browser' is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. The 'Jenkinsfile' field is empty.

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

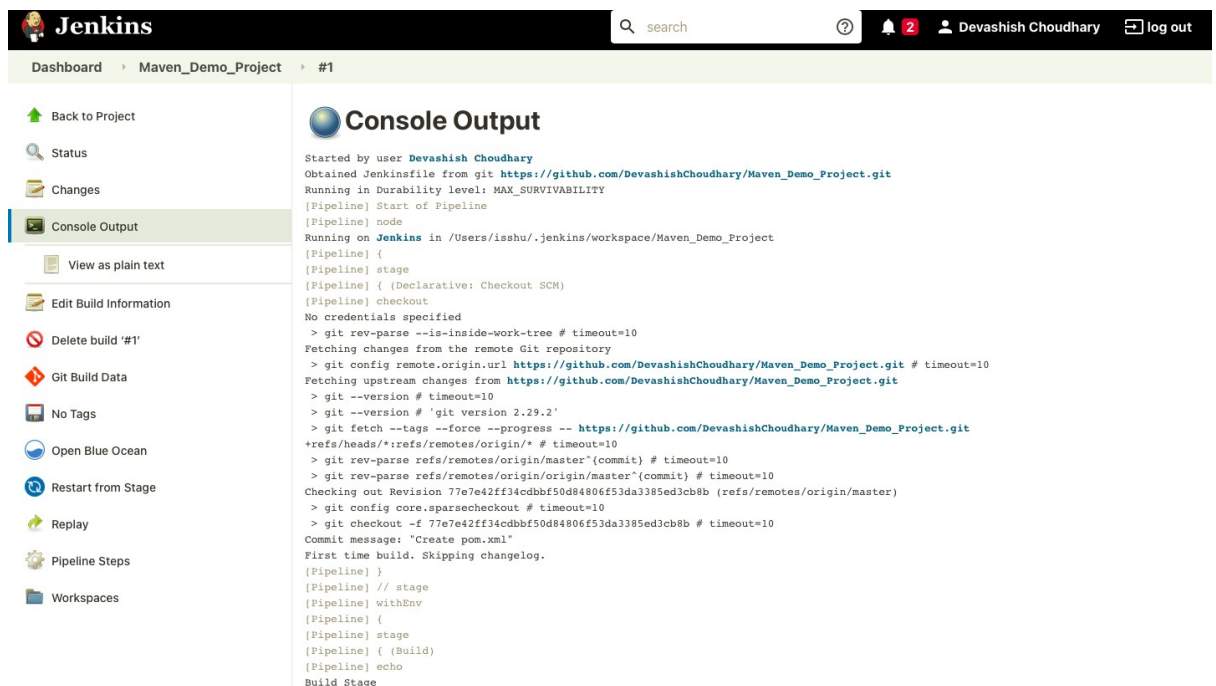


The screenshot shows the Jenkins Pipeline View for the 'Maven\_Demo\_Project'. The left sidebar contains navigation options: Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, and Pipeline Syntax. The main area displays the 'Pipeline Maven\_Demo\_Project' with a 'Recent Changes' section showing a change on Nov 20 at 01:57. Below this is the 'Stage View' showing a table of stage times:

| Declarative: Checkout SCM | Build | Test | Clean |
|---------------------------|-------|------|-------|
| 2s                        | 179ms | 3s   | 161ms |

The 'Permalinks' section is also visible. The bottom right corner shows 'REST API' and 'Jenkins 2.260'.

Console Output as shown below.



The screenshot shows the Jenkins Console Output for the 'Maven\_Demo\_Project'. The left sidebar contains navigation options: Back to Project, Status, Changes, Console Output, View as plain text, Edit Build Information, Delete build '#1', Git Build Data, No Tags, Open Blue Ocean, Restart from Stage, Replay, Pipeline Steps, and Workspaces. The main area displays the 'Console Output' with the following text:

```

Started by user Devashish Choudhary
Obtained Jenkinsfile from git https://github.com/DevashishChoudhary/Maven_Demo_Project.git
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /Users/Isshu/.jenkins/workspace/Maven_Demo_Project
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/DevashishChoudhary/Maven_Demo_Project.git # timeout=10
Fetching upstream changes from https://github.com/DevashishChoudhary/Maven_Demo_Project.git
> git --version # timeout=10
> git --version 'git version 2.29.2'
> git fetch --tags --force --progress -- https://github.com/DevashishChoudhary/Maven_Demo_Project.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 77e7e42ff34cbbf50d84806f53da3385ed3cb8b (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 77e7e42ff34cbbf50d84806f53da3385ed3cb8b # timeout=10
Commit message: "Create pom.xml"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Build Stage
  
```

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

Dashboard
Maven\_Demo\_Project
#1

```

[Pipeline] { (Build)
[Pipeline] echo
Build Stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh
+ mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] -----< demo.project:maven-demo-project >-----
[INFO] Building maven-project 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ maven-demo-project ---
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.738 s
[INFO] Finished at: 2020-11-20T01:57:25+05:30
[INFO]
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Clean)
[Pipeline] echo
Clean Stage
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

REST API
Jenkins 2.260

## Uploading Artifact on Nexus

- Creating repository on Nexus as mention below:

Sonatype Nexus Repository Manager
OSS 3.28.1-01

Administration
Repository

Repository administration

Blob Stores
Cleanup Policies
Repositories
Routing Rules
Content Selectors

Repositories
Blob Stores
Content Selectors
Cleanup Policies
Routing Rules
Security
Privileges
Roles
Users
Anonymous Access
Realms
SSL Certificates
Support
System

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



**Sonatype Nexus Repository Manager** OSS 3.28.1-01

Administration / **Repositories** / Select Recipe / Create Repository: maven2 (hosted)

**Name:** A unique identifier for this repository  
Maven\_Demo\_Project

**Online:** ☒ If checked, the repository accepts incoming requests

**Maven 2**

**Version policy:**  
What type of artifacts does this repository store?  
Snapshot

**Layout policy:**  
Validate that all paths are maven artifact or metadata paths  
Strict

**Storage**

**Blob store:**  
Blob store used to store repository contents  
default

**Strict Content Type Validation:**  
☒ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

**Hosted**

**Deployment policy:**  
Controls if deployments of and updates to artifacts are allowed  
Disable redeploy

**Cleanup**

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

**Sonatype Nexus Repository Manager** OSS 3.28.1-01

Administration / **Repositories** Manage repositories

Create repository Filter

| Name ↑             | Type   | Format | Status                    | URL                    |
|--------------------|--------|--------|---------------------------|------------------------|
| maven-central      | proxy  | maven2 | Online - Ready to Connect | <a href="#">copy</a> > |
| maven-public       | group  | maven2 | Online                    | <a href="#">copy</a> > |
| maven-releases     | hosted | maven2 | Online                    | <a href="#">copy</a> > |
| maven-snapshots    | hosted | maven2 | Online                    | <a href="#">copy</a> > |
| Maven_Demo_Project | hosted | maven2 | Online                    | <a href="#">copy</a> > |

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

```
<distributionManagement>
  <snapshotRepository>
    <id>Maven_Demo_Project</id>
    <name>Maven_Demo_Project</name>
    <url>http://localhost:8081/repository/Maven_Demo_Project/</url>
  </snapshotRepository>
</distributionManagement>
```

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

```
<server>
  <id>Maven_Demo_Project</id>
  <username>admin</username>
  <password>admin</password>
</server>
```

- See the following console output with different command:

- mvn clean

```
maven-demo-project — zsh — 80x24
issu@Devashishs-MacBook-Air maven-demo-project % mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] -----< demo.project:maven-demo-project >-----
[INFO] Building maven-project 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ maven-demo-project ---
[INFO] Deleting /Users/issu/git/Maven_Demo_Project/maven-demo-project/target
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 0.567 s
[INFO] Finished at: 2020-11-20T11:46:53+05:30
[INFO] -----
issu@Devashishs-MacBook-Air maven-demo-project %
```

- mvn install

```
issu@Devashishs-MacBook-Air maven-demo-project % mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] -----< demo.project:maven-demo-project >-----
[INFO] Building maven-project 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ maven-demo-project ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform
m dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.6.0:compile (default-compile) @ maven-demo-project ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ maven-demo-project ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform
m dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.6.0:testCompile (default-testCompile) @ maven-demo-project ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform depe
ndent!
[INFO] Compiling 1 source file to /Users/issu/git/Maven_Demo_Project/maven-demo-project/target/test
-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ maven-demo-project ---
[INFO] Surefire report directory: /Users/issu/git/Maven_Demo_Project/maven-demo-project/target/sure
fire-reports
```

- mvn deploy

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