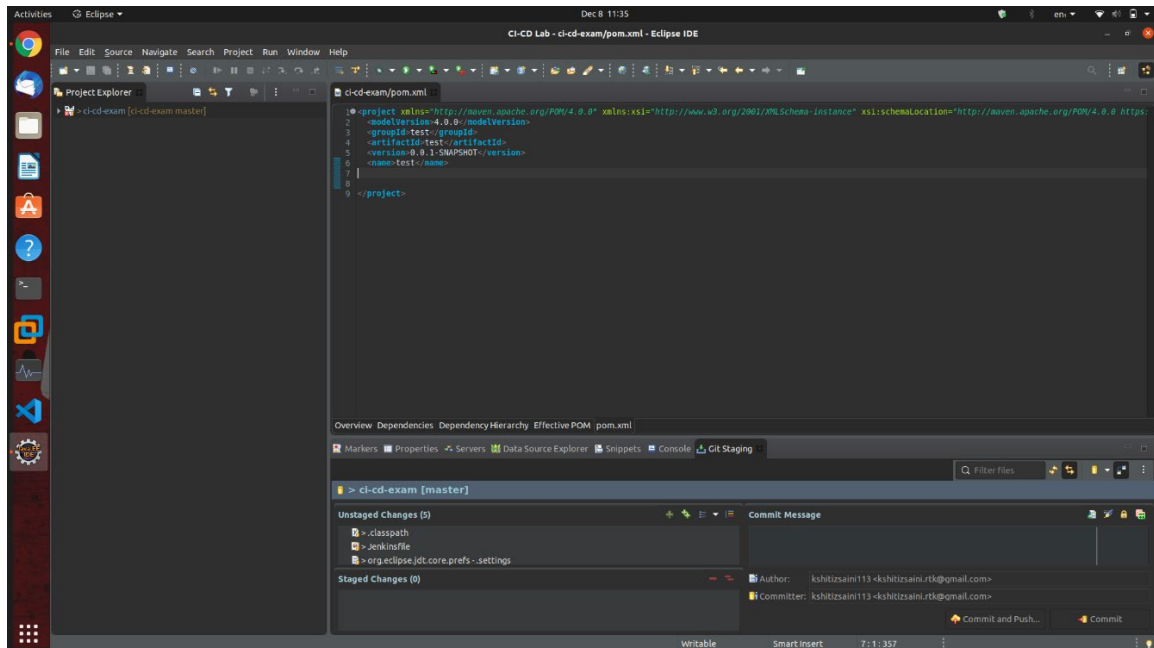


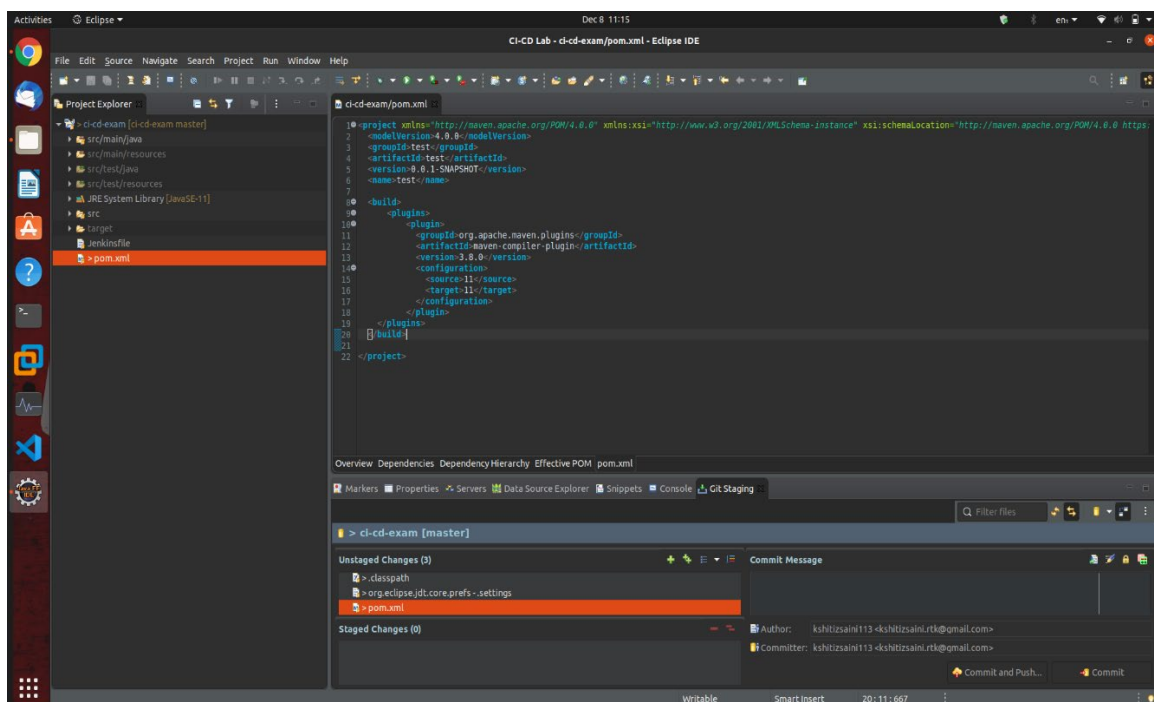
CI-CD Lab Exam

Creating a Jenkins Pipeline

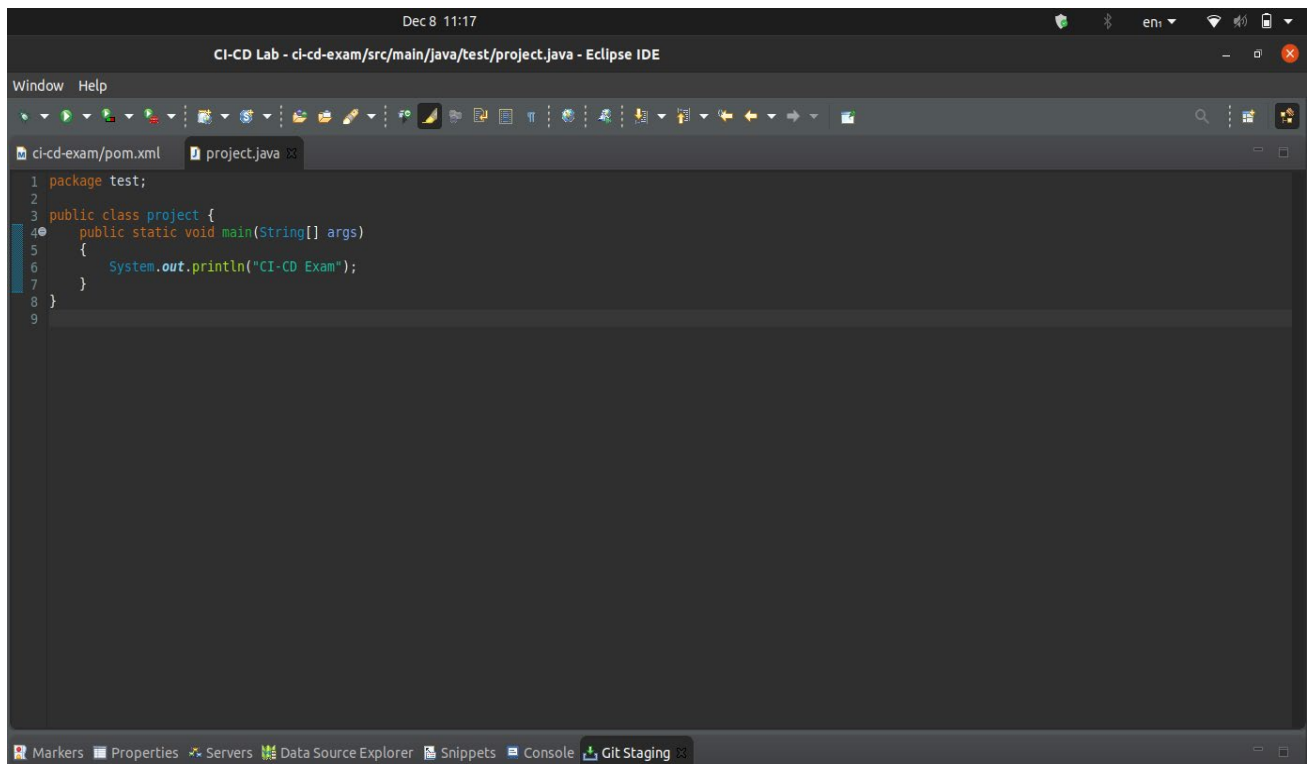
- 1) First of all, we have to create a Maven project to maintain a Jenkins Pipeline for.



- 2) Now we need to add the Maven and Java versions in our POM.xml file



3) Create a simple Java class and a simple program.

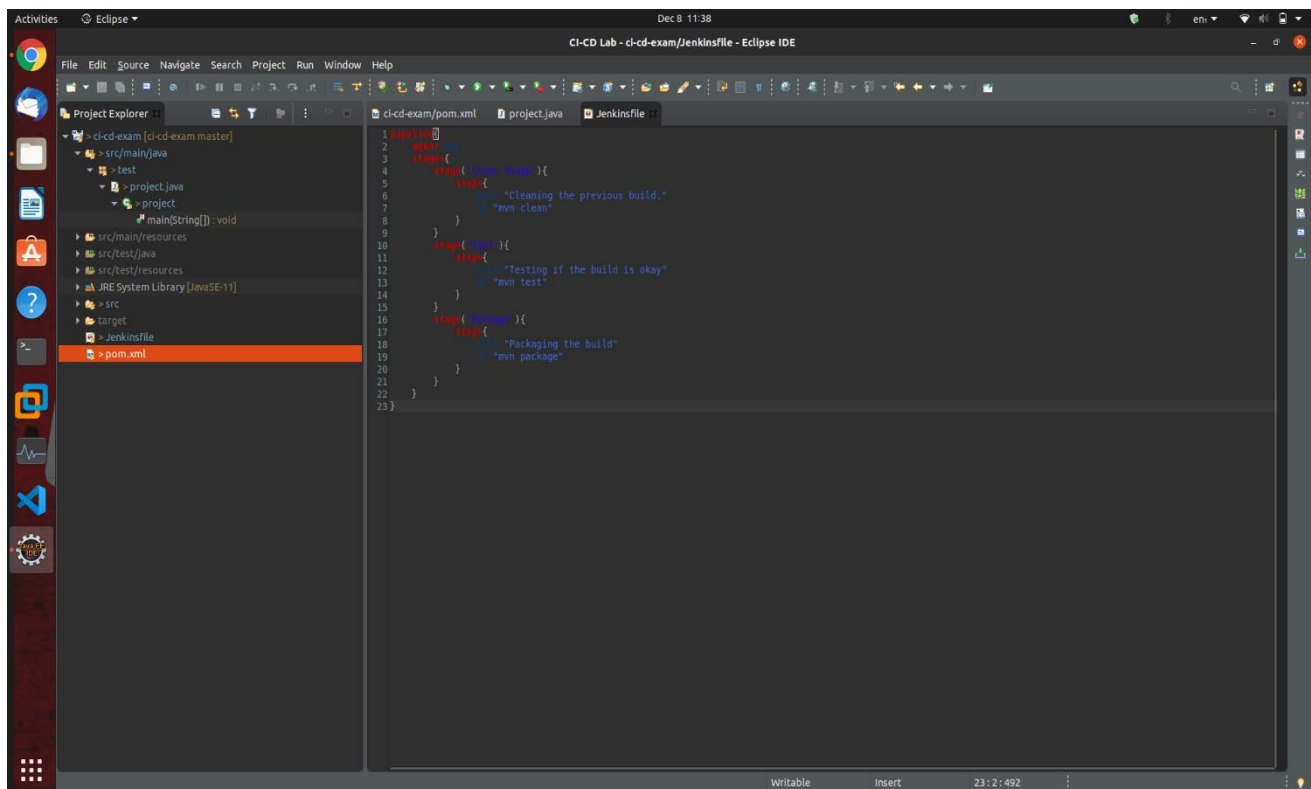


The screenshot shows the Eclipse IDE interface. The title bar indicates the file path: `CI-CD Lab - ci-cd-exam/src/main/java/test/project.java - Eclipse IDE`. The main editor window displays the following Java code:

```
1 package test;
2
3 public class project {
4     public static void main(String[] args)
5     {
6         System.out.println("CI-CD Exam");
7     }
8 }
9
```

The bottom of the IDE shows a toolbar with icons for Markers, Properties, Servers, Data Source Explorer, Snippets, Console, and Git Staging.

4) Create a desired JenkinsFile to achieve the automation pipeline.



The screenshot shows the Eclipse IDE interface with the `Jenkinsfile` open. The title bar indicates the file path: `CI-CD Lab - ci-cd-exam/Jenkinsfile - Eclipse IDE`. The main editor window displays the following Jenkinsfile code:

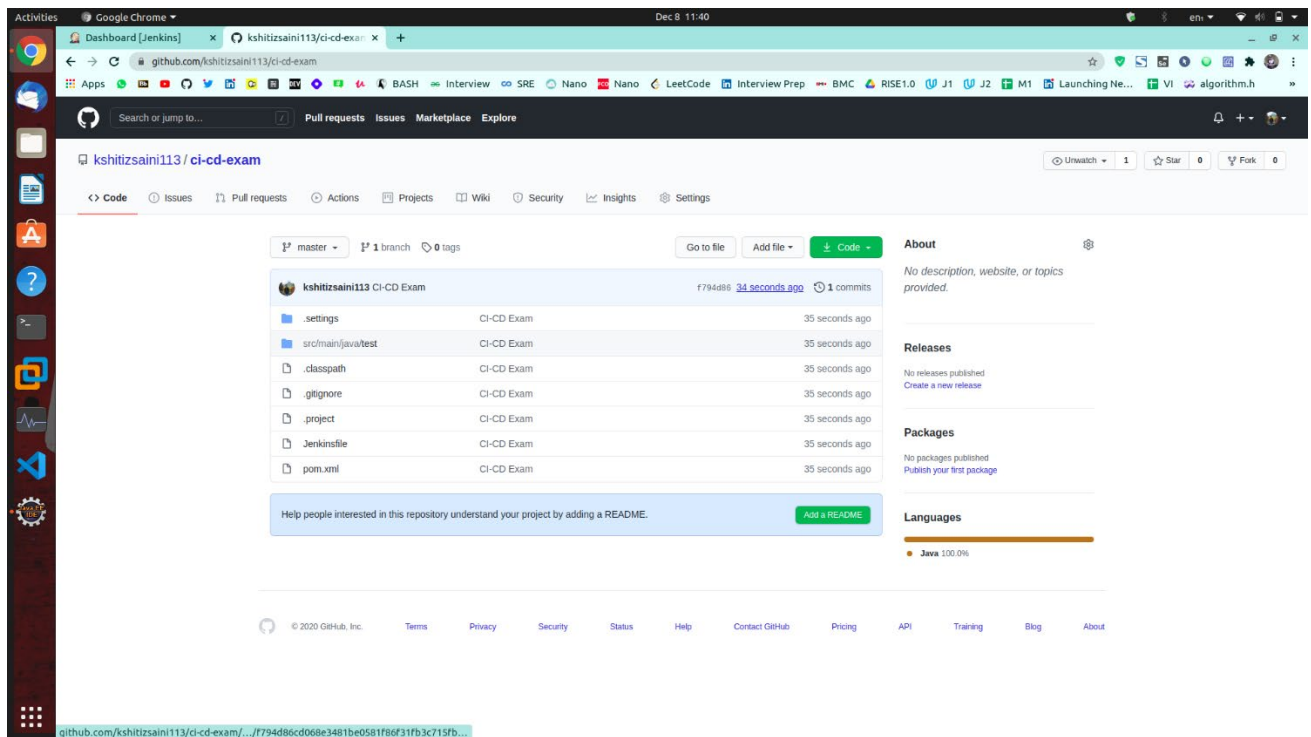
```
1 pipeline{
2     agent any
3     stages{
4         stage('Clean Stage'){
5             steps{
6                 "Cleaning the previous build."
7                 "mvn clean"
8             }
9         }
10        stage('Test'){
11            steps{
12                "Testing if the build is okay"
13                "mvn test"
14            }
15        }
16        stage('Package'){
17            steps{
18                "Packaging the build"
19                "mvn package"
20            }
21        }
22    }
23}
```

The left sidebar shows the Project Explorer with the following structure:

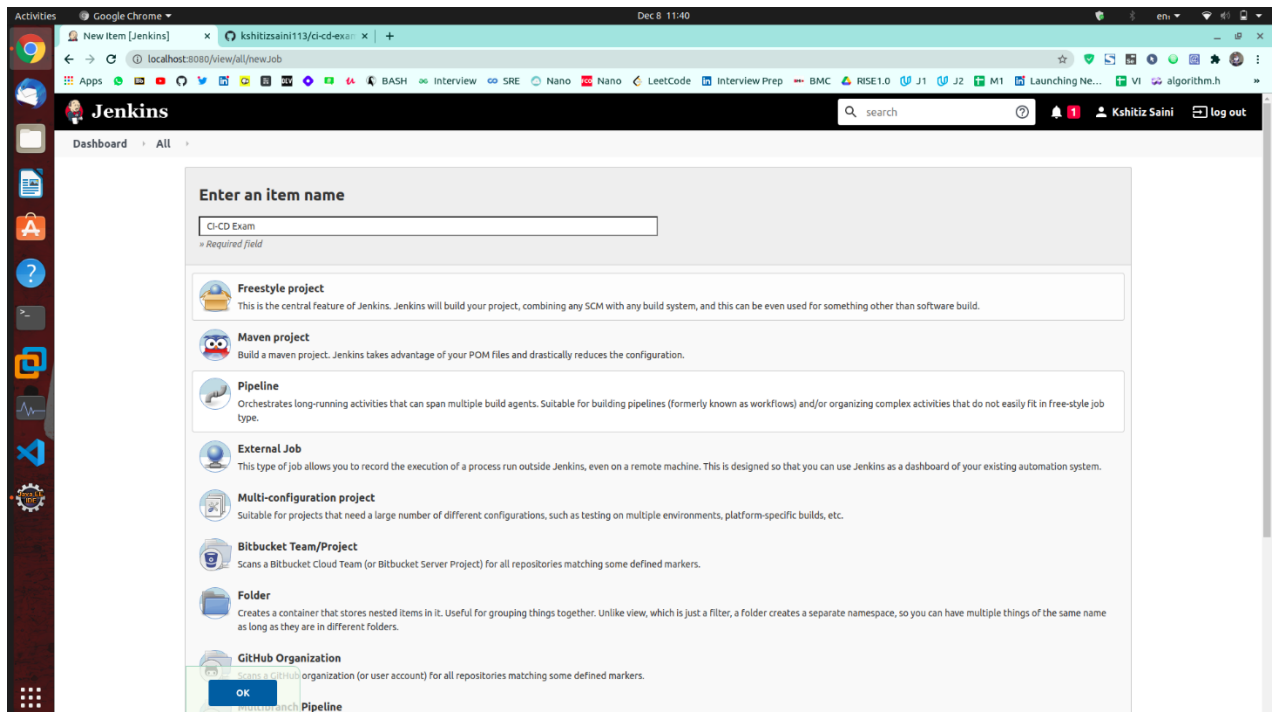
- ci-cd-exam [ci-cd-exam master]
 - src/main/java
 - test
 - project.java
 - project
 - main(String[]): void
 - src/main/resources
 - src/test/java
 - src/test/resources
 - JRE System Library [JavaSE-11]
 - src
 - target
 - Jenkinsfile
 - pom.xml

The bottom status bar shows the text "Writable Insert 23:2:492".

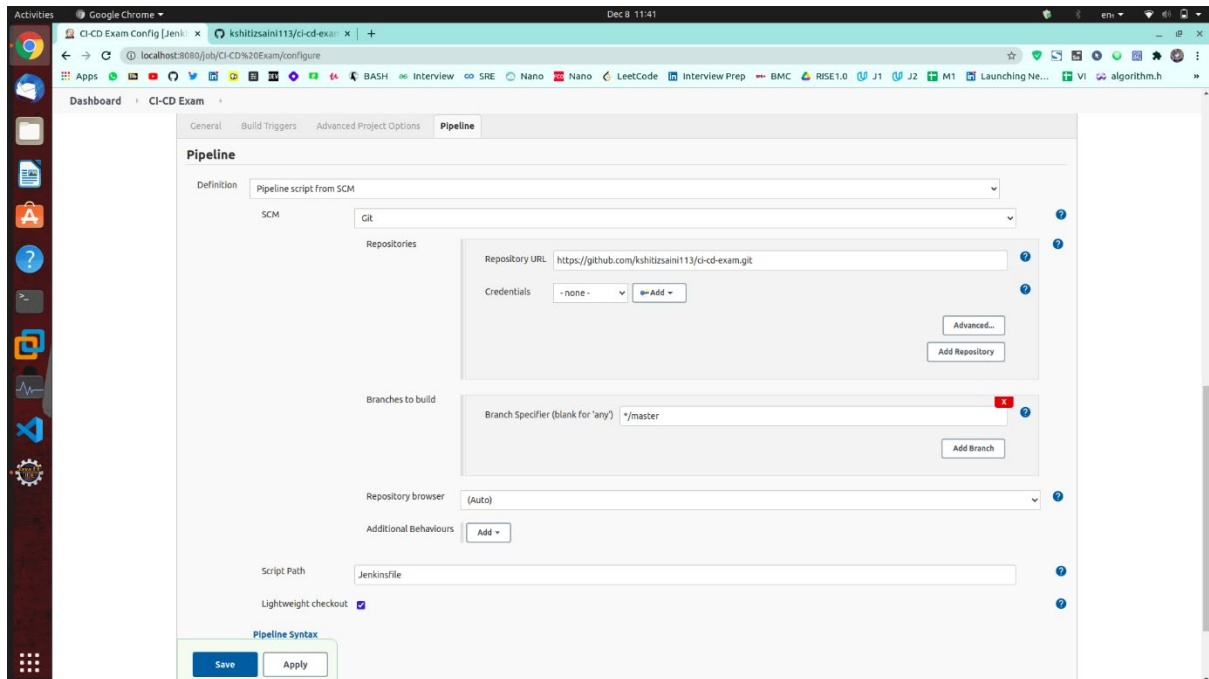
5) Upload the file to GitHub.



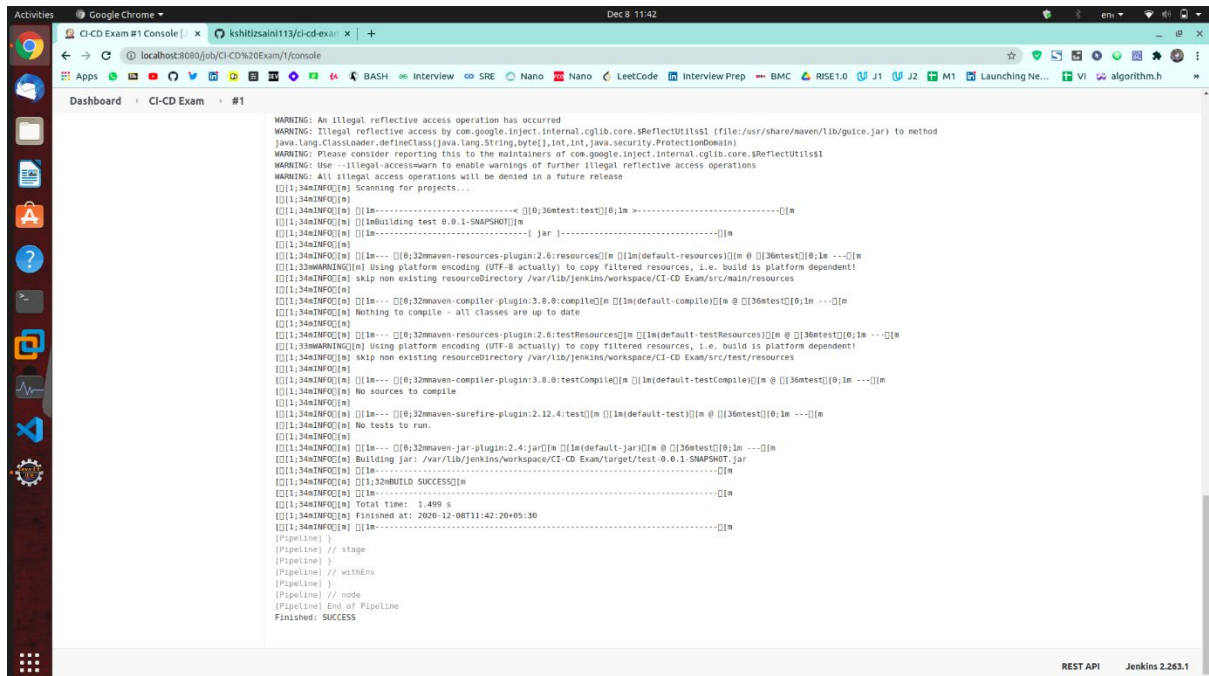
6) Create a new pipeline.



7) Link the pipeline to the GitHub repository.



8) Run the build to see the updates.



9) See the build status.

The screenshot shows the Jenkins web interface for a pipeline named "CI-CD Exam". The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, Pipeline Syntax, Build History, and a search bar. The main content area displays the "Pipeline CI-CD Exam" dashboard. It includes a "Recent Changes" section, a "Stage View" table, and a "Permalinks" section. The "Stage View" table shows the average stage times for four stages: Declarative: Checkout SCM (2s), Clean Stage (2s), Test (4s), and Package (3s). The "Permalinks" section lists the last build (#1) with a duration of 45 seconds, and the last stable, successful, and completed builds, all 45 seconds ago. The bottom right corner of the page shows "REST API" and "Jenkins 2.263.1".

Pipeline CI-CD Exam

Recent Changes

Stage View

Average stage times:
(Average full run time: ~26s)

	Declarative: Checkout SCM	Clean Stage	Test	Package
Average stage times	2s	2s	4s	3s
Dec 08 11:41	2s	2s	4s	3s

Permalinks

- Last build (#1), 45 sec ago
- Last stable build (#1), 45 sec ago
- Last successful build (#1), 45 sec ago
- Last completed build (#1), 45 sec ago

REST API Jenkins 2.263.1