

# Module 4: Metrics to Improve Quality

---

## Demo Document - 1

**edureka!**

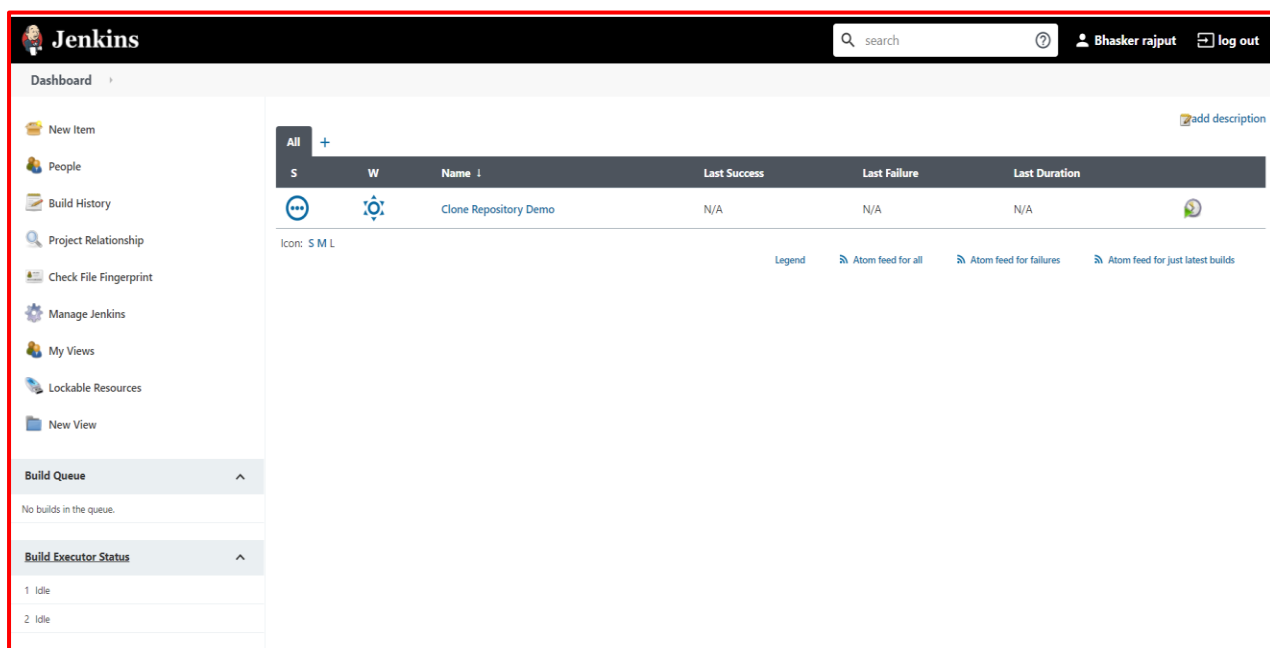
## Demo: Gradle Project with Jenkins.

### Problem Statement:

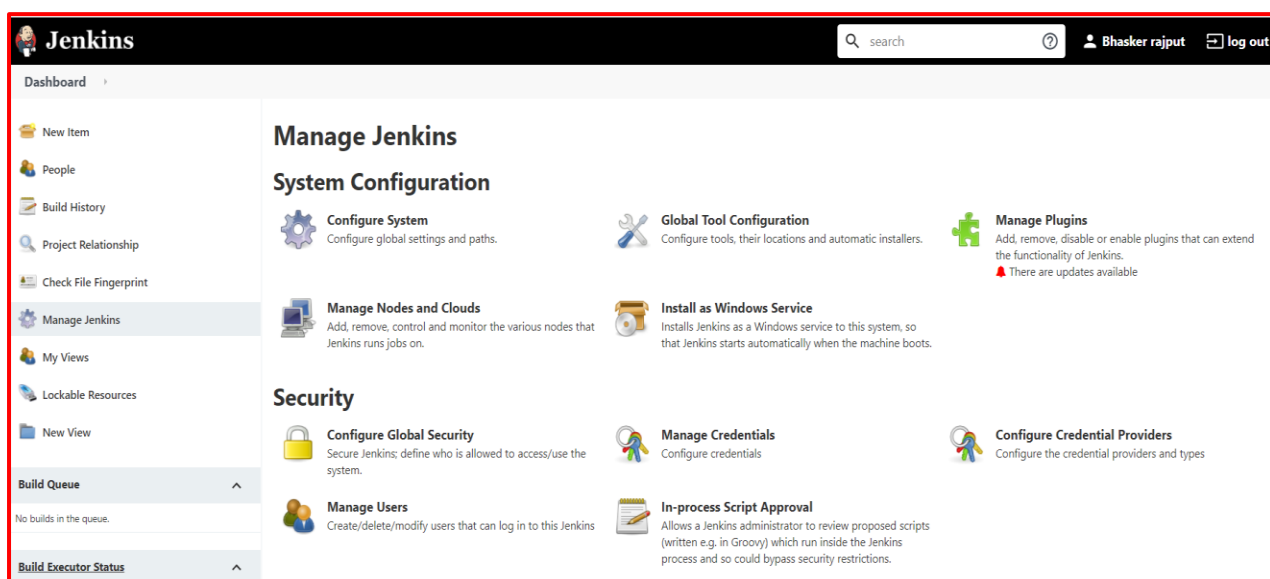
### How to configure and run Code coverage with Jenkins.

### Solution Steps:

1. Login to Jenkins console and click on **‘Manage Jenkins’** from left side menu.



2. Now click on **‘Manage plugins’** to install plugin.



3. Search **JACOCO** in **‘Available’** tab and select it as mentioned in below screenshot. JACOCO is code coverage tool. Now click on **‘Install without restart’**.

The screenshot shows the Jenkins Plugin Manager interface. The top navigation bar includes the Jenkins logo, a search bar, and the user name 'Bhasker rajput'. The breadcrumb trail shows 'Dashboard' > 'Plugin Manager'. On the left sidebar, there are links for 'Back to Dashboard', 'Manage Jenkins', and 'Update Center'. The main content area has a search bar with 'jacoco' entered. Below the search bar are tabs for 'Updates', 'Available', 'Installed', and 'Advanced'. The 'Available' tab is selected, showing a table with columns 'Install', 'Name', 'Version', and 'Released'. The table lists the 'JaCoCo' plugin with version '3.2.0' and release date '1 mo 4 days ago'. Below the table, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. A note indicates 'Update information obtained: 8 min 52 sec ago'.

4. It will take some time to install the '**JACOCO**' plugin and once it is installed successfully click on '**Back to Dashboard**'.

The screenshot shows the Jenkins Update Center interface. The top navigation bar includes the Jenkins logo and the user name 'Bhasker rajput'. The breadcrumb trail shows 'Dashboard' > 'Update Center'. On the left sidebar, there are links for 'Back to Dashboard', 'Manage Jenkins', and 'Manage Plugins'. The main content area has a heading 'Installing Plugins/Upgrades'. Below the heading, there is a 'Preparation' section with a list of steps: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. Below this, there is a list of installed plugins with their status: 'Javadoc' (Success), 'Maven Integration' (Success), 'Loading plugin extensions' (Success), 'JaCoCo' (Success), and 'Loading plugin extensions' (Success). At the bottom, there are two green arrows pointing right, one with the text 'Go back to the top page (you can start using the installed plugins right away)' and another with the text 'Restart Jenkins when installation is complete and no jobs are running'.

5. Now you need to add **'JACOCO' plugin configuration code** inside **'POM.XML'** file of your JAVA MAVEN project. We have already added 'JACOCO' plugin configuration code to below mentioned GITHUB repository which is used as a part of this demo.

**GITHUB URL** - <https://github.com/BhaskerRajput/CodeCoverageDemo.git>

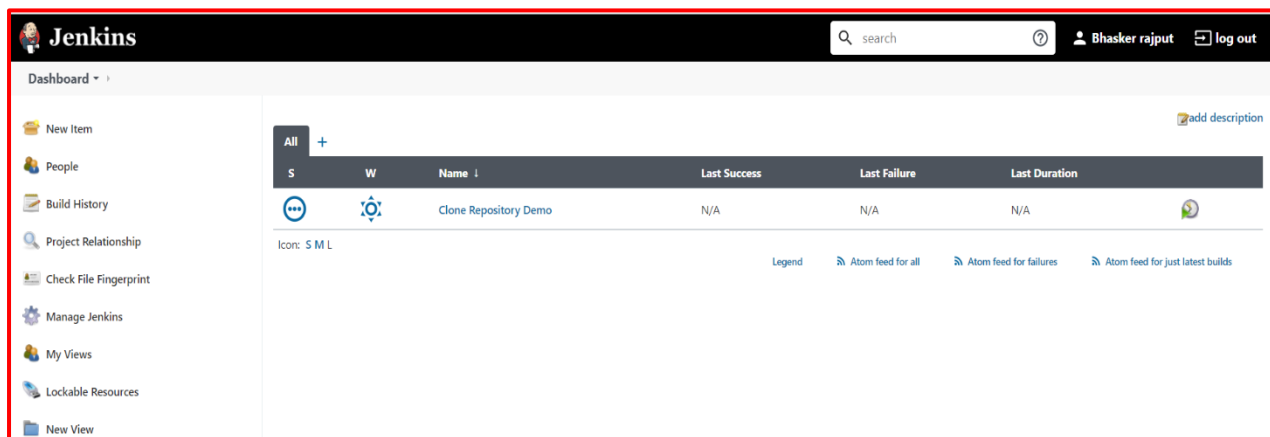
**NOTE** - In case you need to integrate 'JACOCO' plugin configuration code to other JAVA MAVEN project then use the below piece of code and add it under 'Plugin' section of your project's 'POM.XML' file.

```
<plugin>

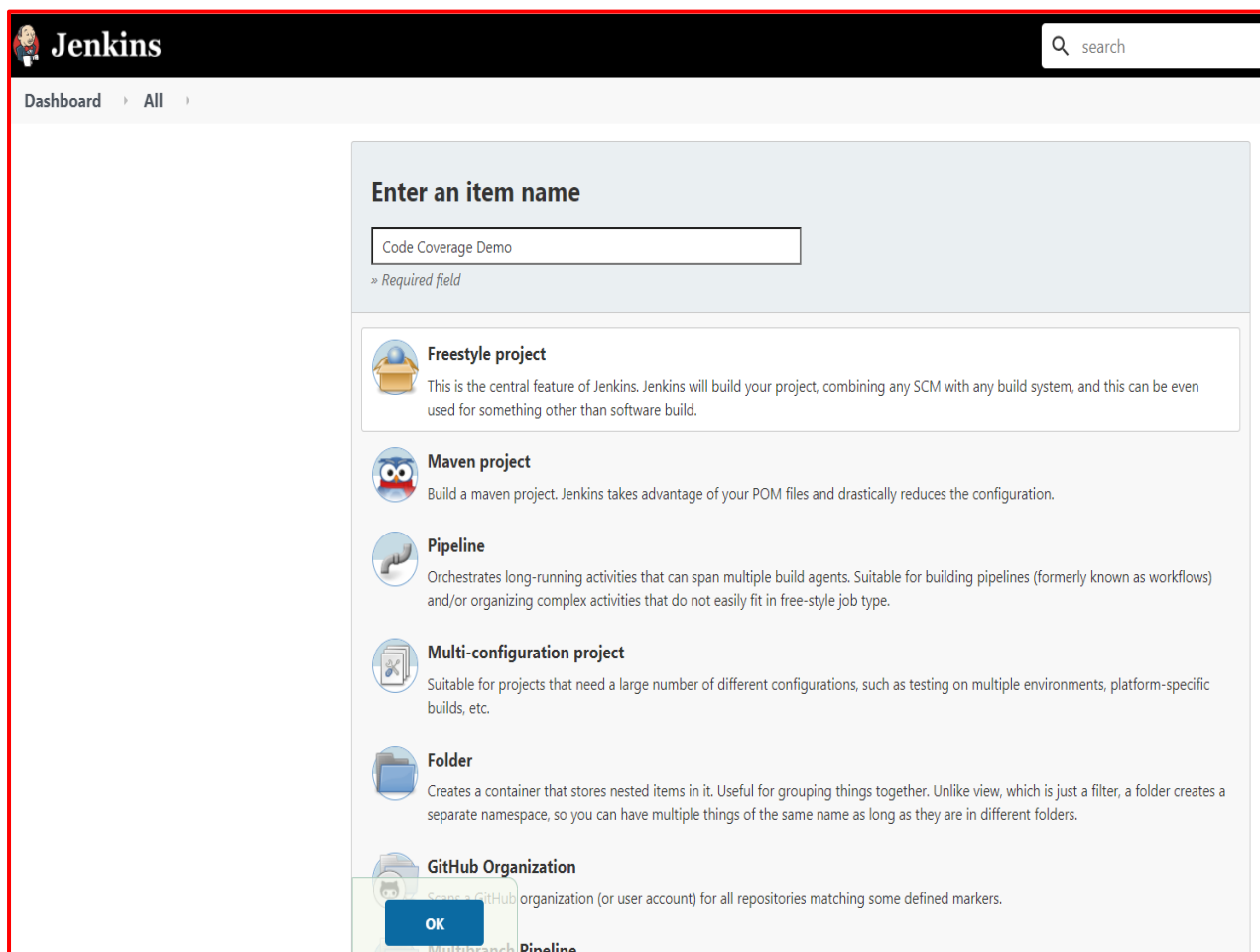
    <groupId>org.jacoco</groupId>
    <artifactId>jacoco-maven-plugin</artifactId>
    <version>0.8.3</version>
    <executions>
        <execution>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>jacoco-report</id>
            <phase>test</phase>
            <goals>
                <goal>report</goal>
            </goals>
            <!-- default target/jscoco/site/* -->
            <configuration>
                <outputDirectory>target/jacoco-
report</outputDirectory>
                <dataFile>${project.build.directory}/coverage-
reports/jacoco.exec</dataFile>
                <rules>
                    <rule>
                        <element>CLASS</element>
                        <excludes>

                            <exclude>com.asimio.demo.Application</exclude>
                        </excludes>
                    </rule>
                </rules>
            </configuration>
        </execution>
    </executions>
</plugin>
```

6. Now click on **'New Item'** to create new Jenkins job.



7. Enter the 'Item Name' as 'Code Coverage Demo' and select 'Freestyle project'. Now click on OK button.



8. Navigate to 'Source Code Management' section and select 'GIT' radio button to enter the below 'Repository URL'.

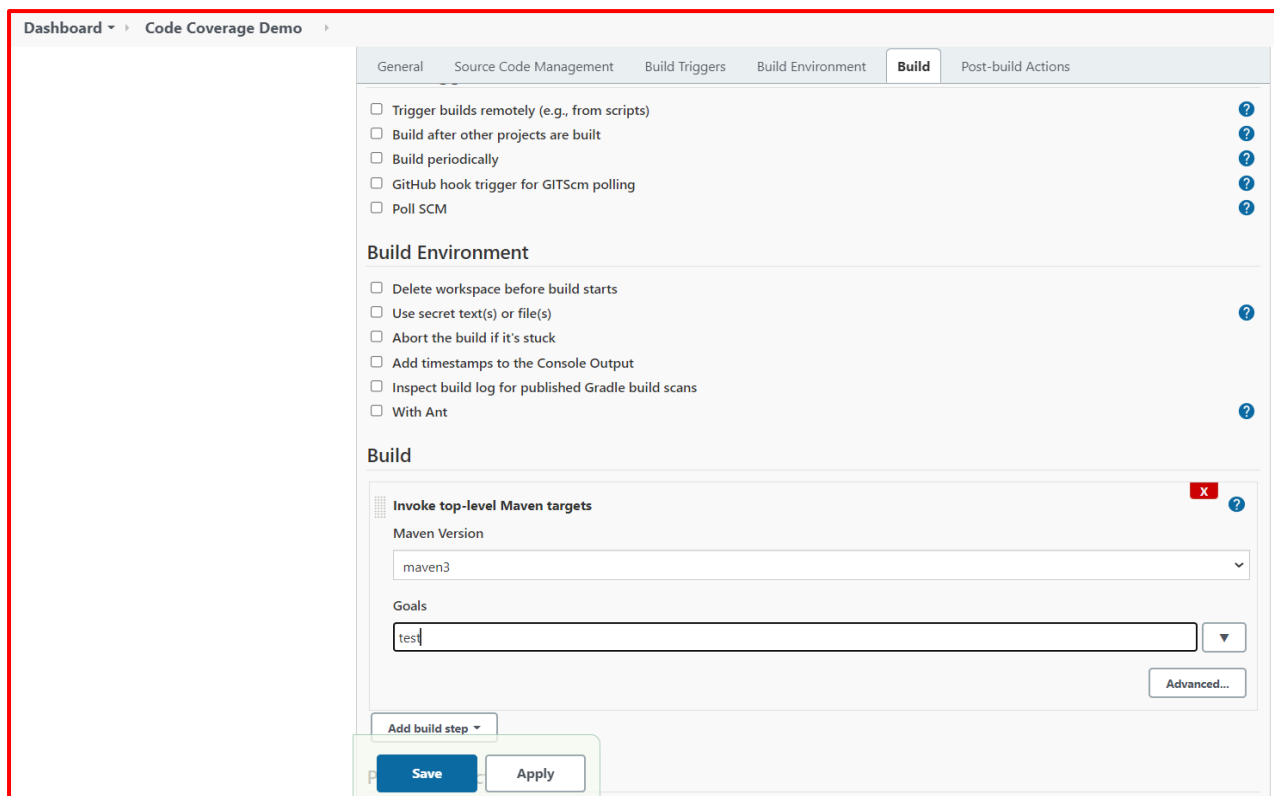
**GITHUB URL - <https://github.com/BhaskerRajput/CodeCoverageDemo.git>**

The screenshot shows the 'Source Code Management' tab of a build system configuration. The 'General' tab is selected, and the 'Git' option is chosen under 'Repositories'. The 'Repository URL' field contains 'https://github.com/BhaskerRajput/CodeCoverageDemo.git'. The 'Credentials' dropdown is set to '- none -'. The 'Branches to build' section has a 'Branch Specifier (blank for 'any')' field with the value '\*/master'. The 'Repository browser' dropdown is set to '(Auto)'. There are 'Save' and 'Apply' buttons at the bottom.

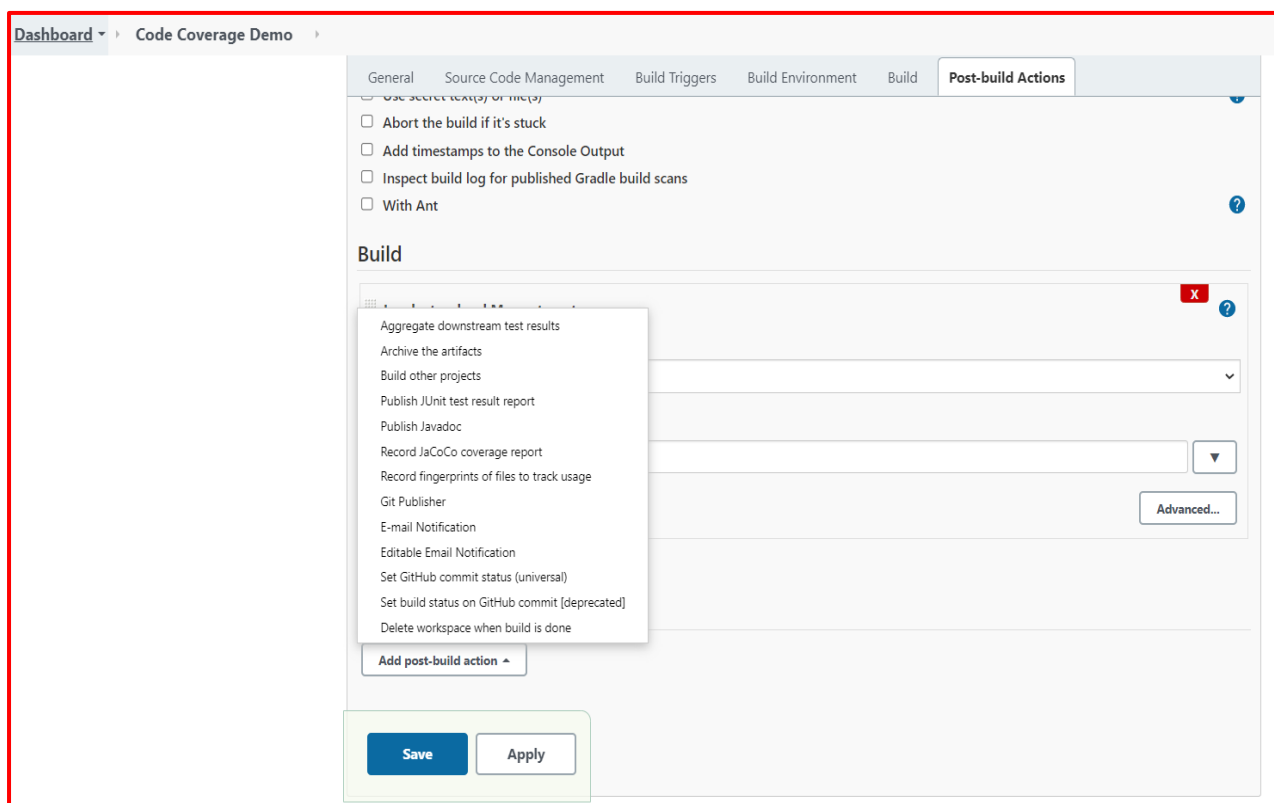
9. Navigate to **'Build'** section and click on **'Add build step'** and select **'Invoke top-level Maven targets'** from the list.

The screenshot shows the 'Build' tab of the build system configuration. The 'Build' section is active, and the 'Add build step' button is clicked, opening a dropdown menu. The menu lists several options: 'Execute Windows batch command', 'Execute shell', 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Run with timeout', and 'Set build status to "pending" on GitHub commit'. The 'Invoke top-level Maven targets' option is highlighted.

10. Now select the **'Maven Version'** and provide **'Goals'** as **'test'**.



11. Navigate to 'Post-builds Actions' and click on 'Add post-build action' to select 'Record Jacoco coverage report' from the list.



12. Keep configuration as default under 'Record Jacoco coverage report' and click on **APPLY** and **SAVE**.

The screenshot shows the 'Post-build Actions' configuration page for a Jenkins job named 'Code Coverage Demo'. The 'Record JaCoCo coverage report' action is selected. The configuration includes fields for 'Path to exec files', 'Inclusions', 'Exclusions', 'Path to class directories', 'Path to source directories', 'Inclusions', and 'Exclusions'. There are also checkboxes for 'Disable display of source files for coverage', 'Change build status according to the defined thresholds', and 'Always run coverage collection, even if build is FAILED or ABORTED'. At the bottom, there is a table for 'Delta thresholds' with columns for 'Instruction', '% Branch', '% Complexity', '% Line', '% Method', and '% Class'. The 'Save' and 'Apply' buttons are visible at the bottom left.

**Record JaCoCo coverage report**

Path to exec files (e.g.: `**/target/**/*.exec, **/jacoco.exec`)  
 Inclusions (e.g.: `**/*.class`)  
 Exclusions (e.g.: `**/*Test*.class`)

Path to class directories (e.g.: `**/target/classDir, **/classes`)

Path to source directories (e.g.: `**/mySourceFiles`)  
 Inclusions (e.g.: `**/*.java, **/*.groovy, **/*.gs`)  
 Exclusions (e.g.: `generated/**/*.java`)

☐ Disable display of source files for coverage

☐ Change build status according to the defined thresholds

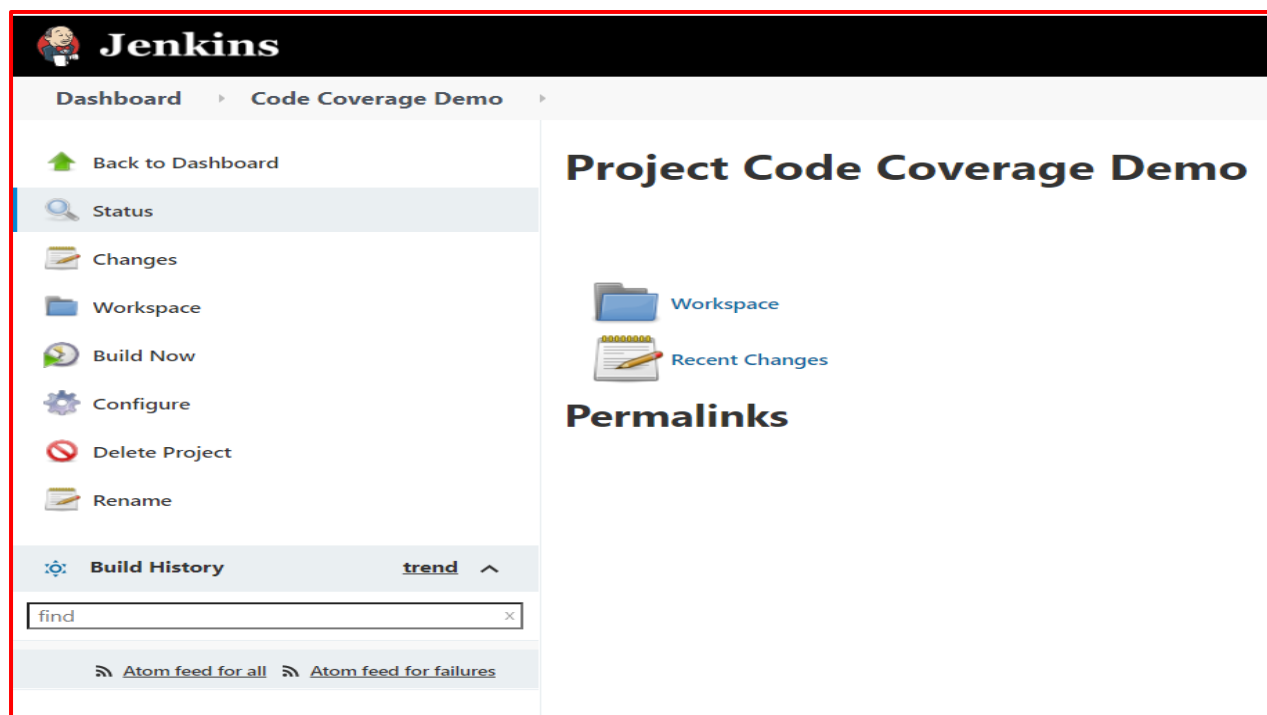
☐ Always run coverage collection, even if build is FAILED or ABORTED

	Instruction	% Branch	% Complexity	% Line	% Method	% Class
☀	0	0	0	0	0	0
☁	0	0	0	0	0	0

☐ Fail the build if coverage degrades more than the delta thresholds

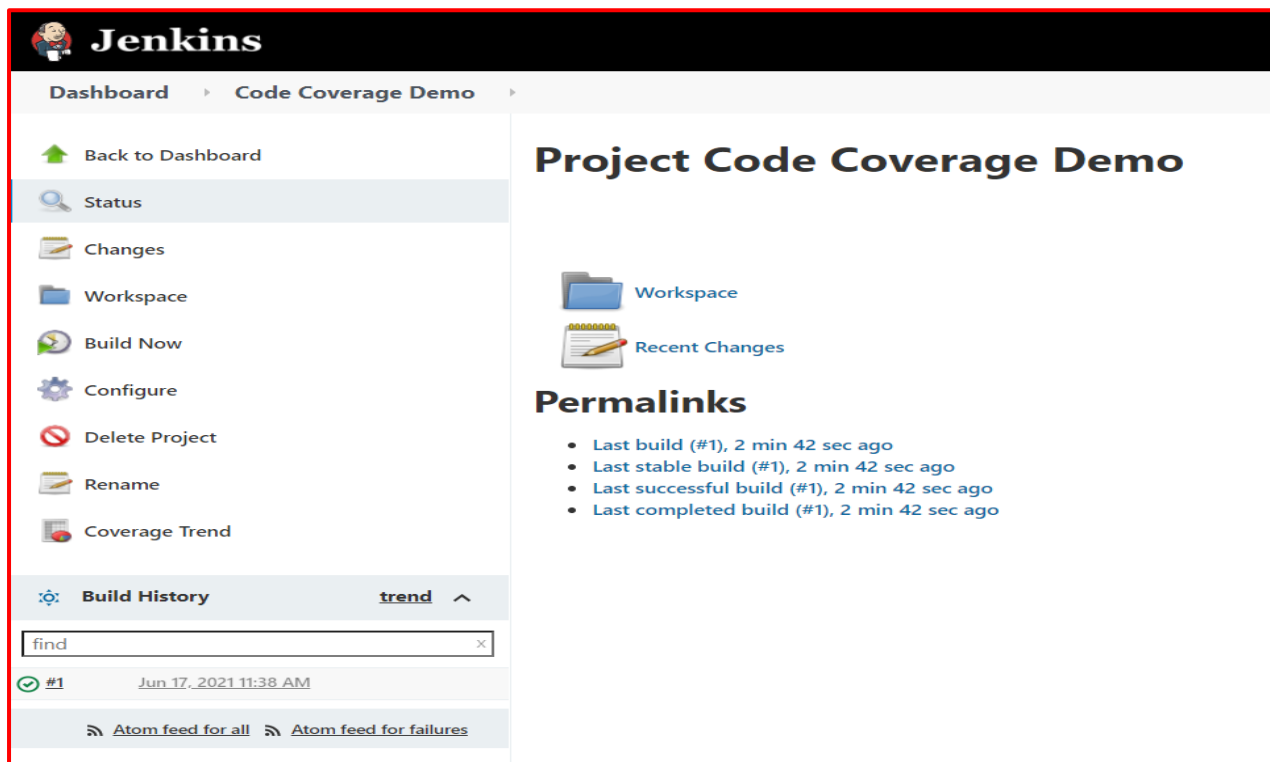
**Save** **Apply**

13. Now build your Jenkins job using 'Build Now' button as shown in below screenshot.

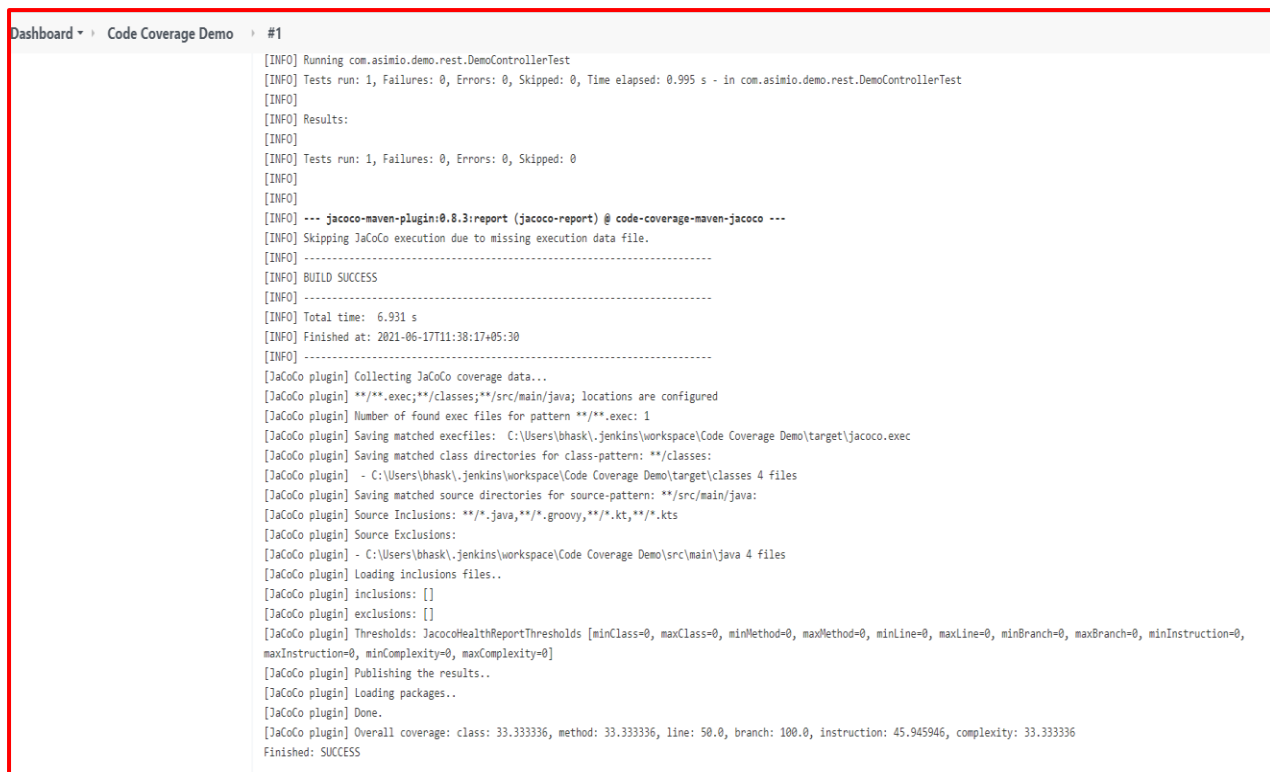


14. Now under 'Build History' select the 'Build number'. Now further select the 'Console output' option to check the logs.



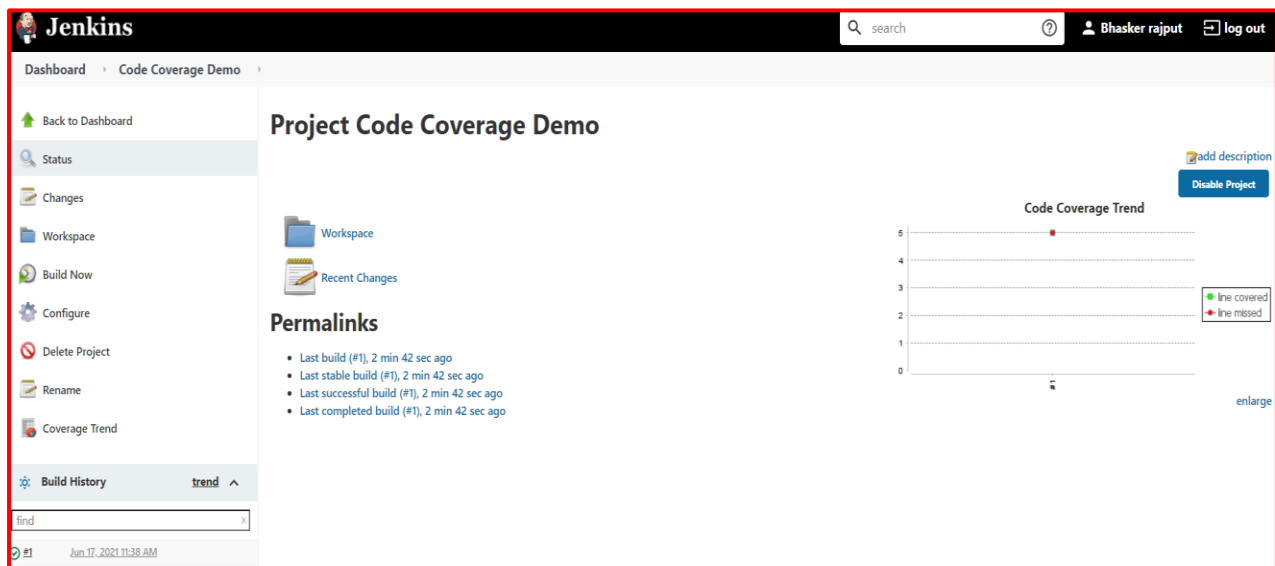


15. Scroll at the bottom of ‘**Console output**’ logs and you can see **SUCCESS** message in logs as shown in the below screenshot.

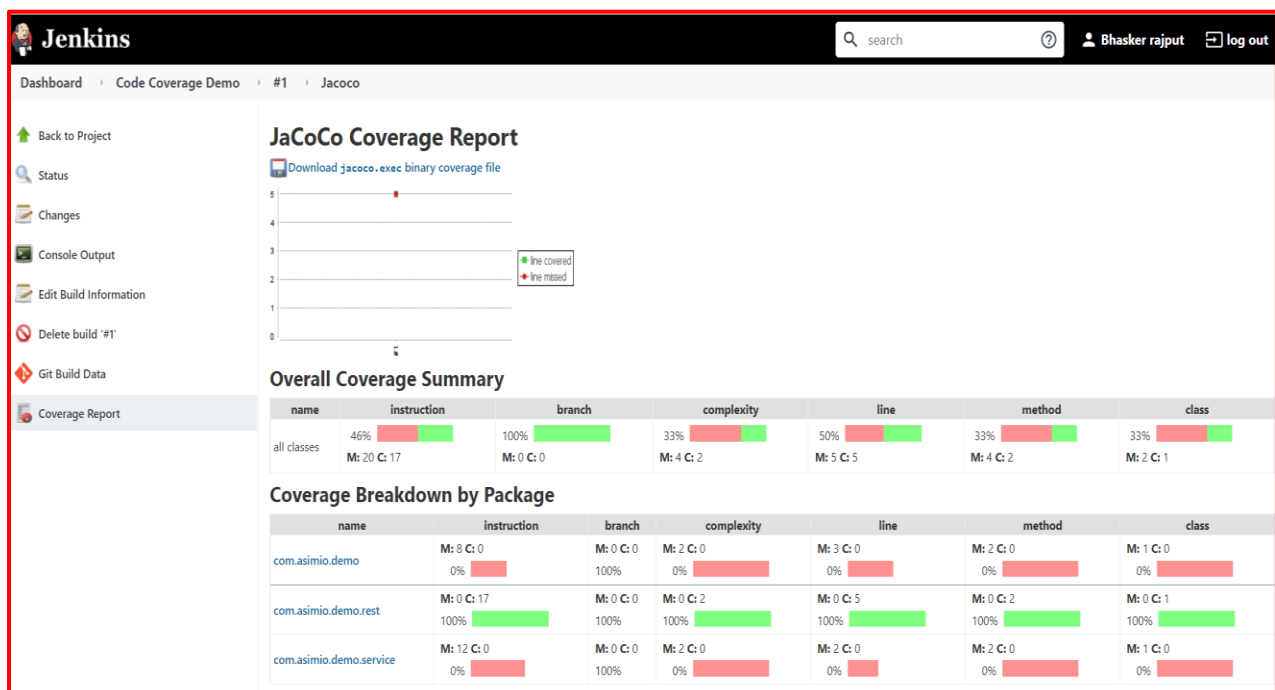


```
[INFO] Running com.asimio.demo.rest.DemoControllerTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.995 s - in com.asimio.demo.rest.DemoControllerTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.3:report (jacoco-report) @ code-coverage-maven-jacoco ---
[INFO] Skipping JaCoCo execution due to missing execution data file.
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 6.931 s
[INFO] Finished at: 2021-06-17T11:38:17+05:30
[INFO]
[JaCoCo plugin] Collecting JaCoCo coverage data...
[JaCoCo plugin] **/*.exec;/**/*.classes;/**/*.src/main/java; locations are configured
[JaCoCo plugin] Number of found exec files for pattern **/*.exec: 1
[JaCoCo plugin] Saving matched execfiles: C:\Users\bhask\jenkins\workspace\Code Coverage Demo\target\jacoco.exec
[JaCoCo plugin] Saving matched class directories for class-pattern: **/*.classes:
[JaCoCo plugin] - C:\Users\bhask\jenkins\workspace\Code Coverage Demo\target\classes 4 files
[JaCoCo plugin] Saving matched source directories for source-pattern: **/*.src/main/java:
[JaCoCo plugin] Source Inclusions: **/*.java,**/*.groovy,**/*.kt,**/*.kts
[JaCoCo plugin] Source Exclusions:
[JaCoCo plugin] - C:\Users\bhask\jenkins\workspace\Code Coverage Demo\src/main/java 4 files
[JaCoCo plugin] Loading inclusions files..
[JaCoCo plugin] inclusions: []
[JaCoCo plugin] exclusions: []
[JaCoCo plugin] Thresholds: JacocoHealthReportThresholds [minClass=0, maxClass=0, minMethod=0, maxMethod=0, minLine=0, maxLine=0, minBranch=0, maxBranch=0, minInstruction=0, maxInstruction=0, minComplexity=0, maxComplexity=0]
[JaCoCo plugin] Publishing the results..
[JaCoCo plugin] Loading packages..
[JaCoCo plugin] Done.
[JaCoCo plugin] Overall coverage: class: 33.333336, method: 33.333336, line: 50.0, branch: 100.0, instruction: 45.945946, complexity: 33.333336
Finished: SUCCESS
```

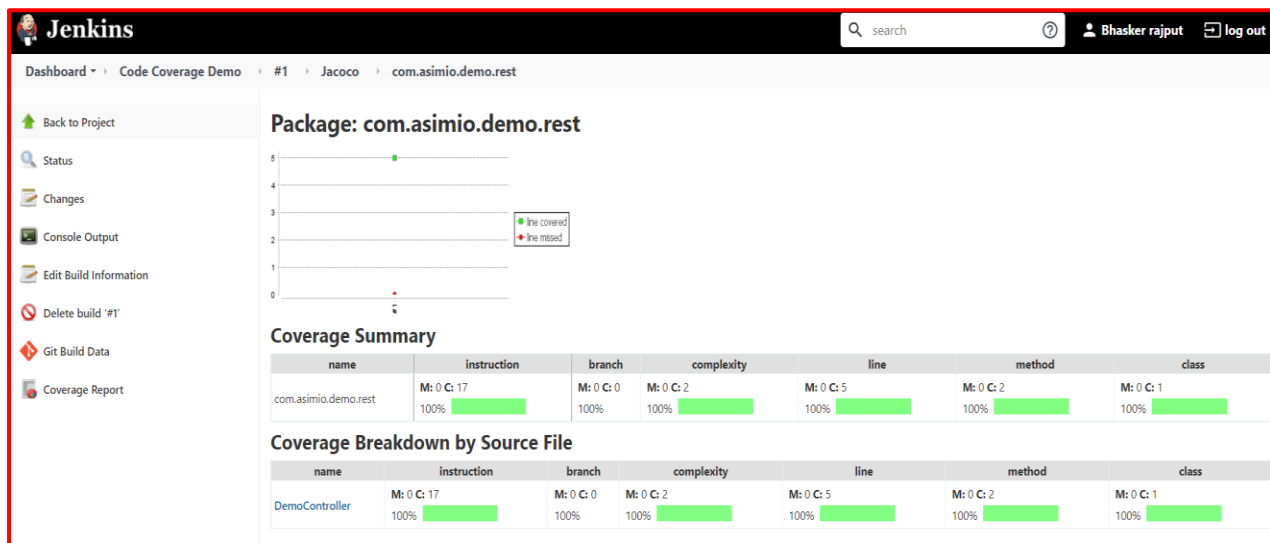
16. Navigate back to ‘**Code Coverage Demo**’ project and click on ‘**Code coverage trend**’ on the right side as shown in the below screenshot.



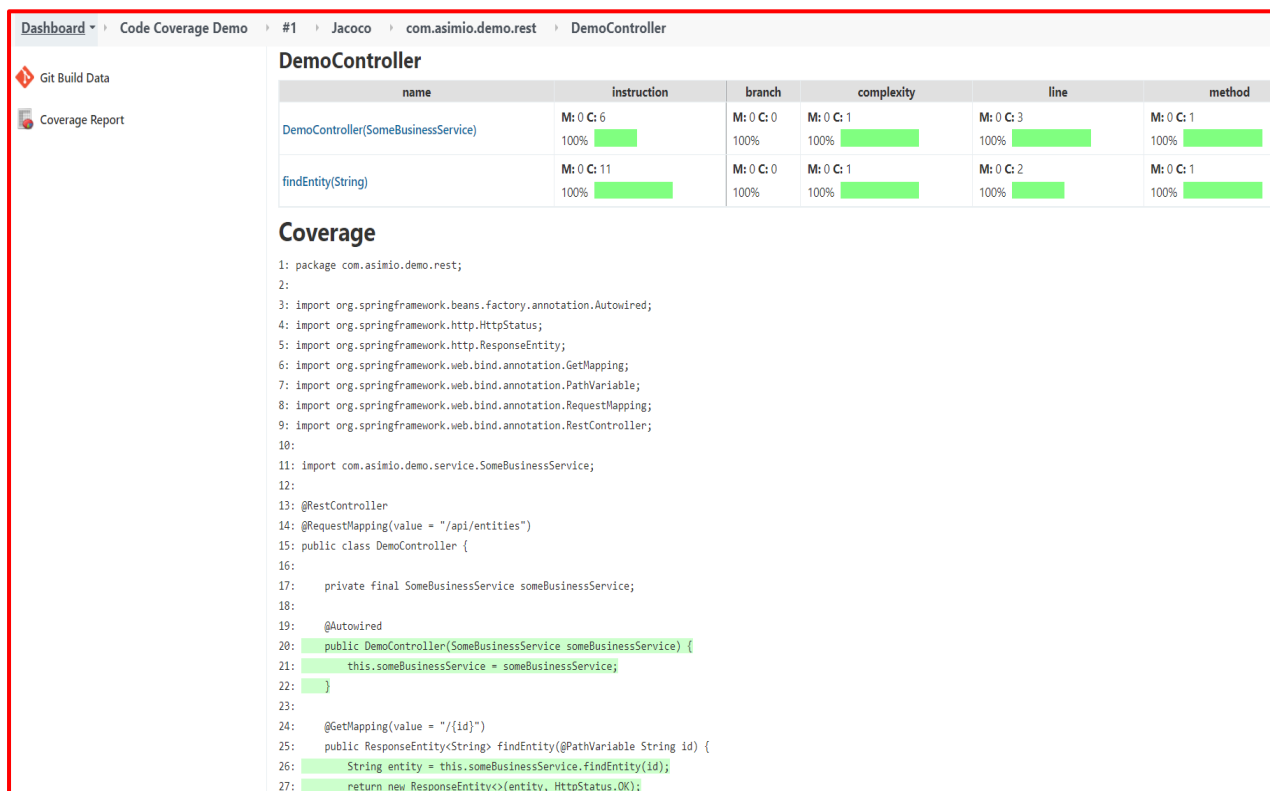
17. Now you can 'Jacoco coverage report'. **GREEN** colour report tells which all code part is covered under test cases and tested successfully while **RED** colour report tells which all code part is not covered under test cases. Now click on '[com.asimio.demo.rest](#)'.



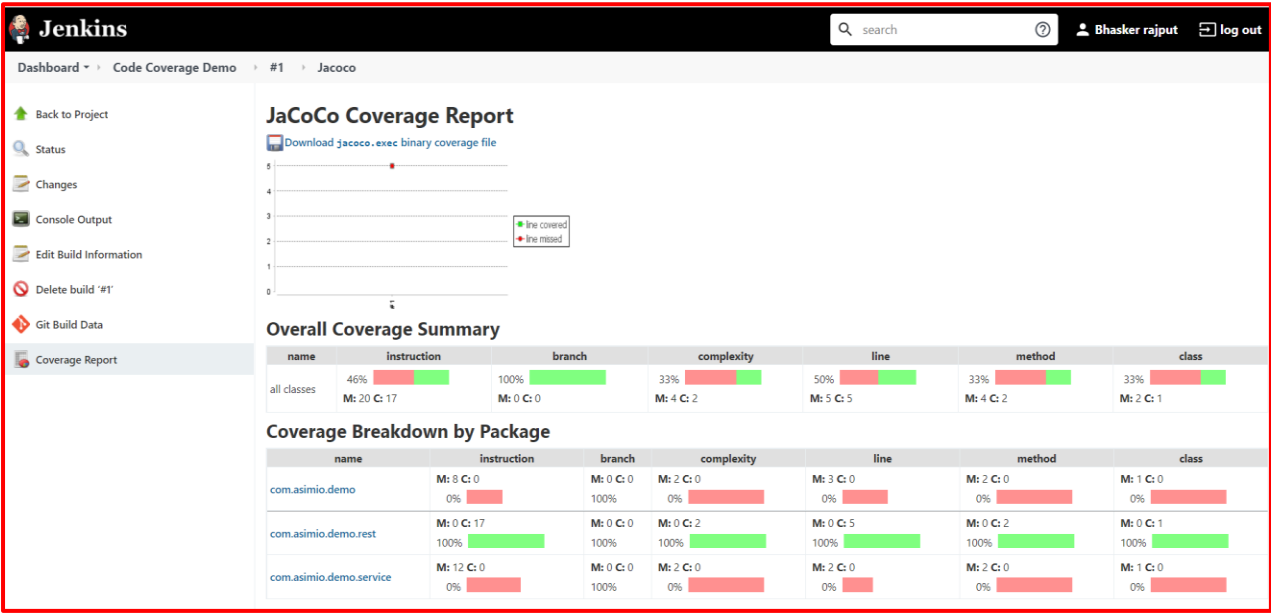
18. Now click on 'DemoController' to check the code coverage.



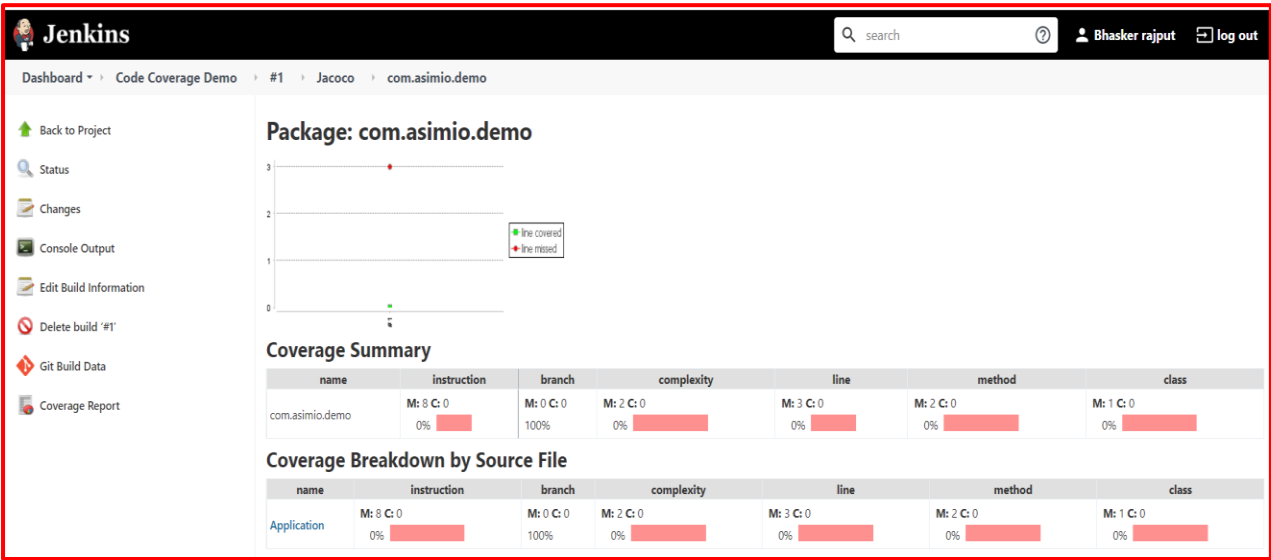
19. Now scroll down and you will see code highlighted in **GREEN** colour which is covered under **JACOCO** code coverage.



20. Now navigate back to ‘**Jacoco coverage report**’ and click on ‘**com.asimio.demo**’.



21. Now click on ‘Application’ to check the code coverage.



22. Now scroll down and you will see code highlighted in **RED** colour which is not covered under JACOCO code coverage.

Dashboard

Code Coverage Demo

#1

Jacoco

com.asimio.demo

Application

Status

Changes

Console Output

Edit Build Information

Delete build '#1'

Git Build Data

Coverage Report

3

2

1

0

line covered

line missed

Application

name	instruction	branch	complexity	line	method
Application()	M: 3 C: 0 0% <div></div>	M: 0 C: 0 100%	M: 1 C: 0 0% <div></div>	M: 1 C: 0 0% <div></div>	M: 1 C: 0 0% <div></div>
main(String[])	M: 5 C: 0 0% <div></div>	M: 0 C: 0 100%	M: 1 C: 0 0% <div></div>	M: 2 C: 0 0% <div></div>	M: 1 C: 0 0% <div></div>

Coverage

```
1: package com.asimio.demo;
2:
3: import org.springframework.boot.SpringApplication;
4: import org.springframework.boot.autoconfigure.SpringBootApplication;
5:
6: @SpringBootApplication
7: public class Application {
8:
9:     public static void main(String[] args) {
10:         SpringApplication.run(Application.class, args);
11:     }
12: }
```