Git-Hub URL

<https://github.com/paragpatil7/API_Automation.git>

Docker hub URL

<https://hub.docker.com/u/paragpatil7/>

Execution can be done in 2 ways

1. **Using Eclipses**

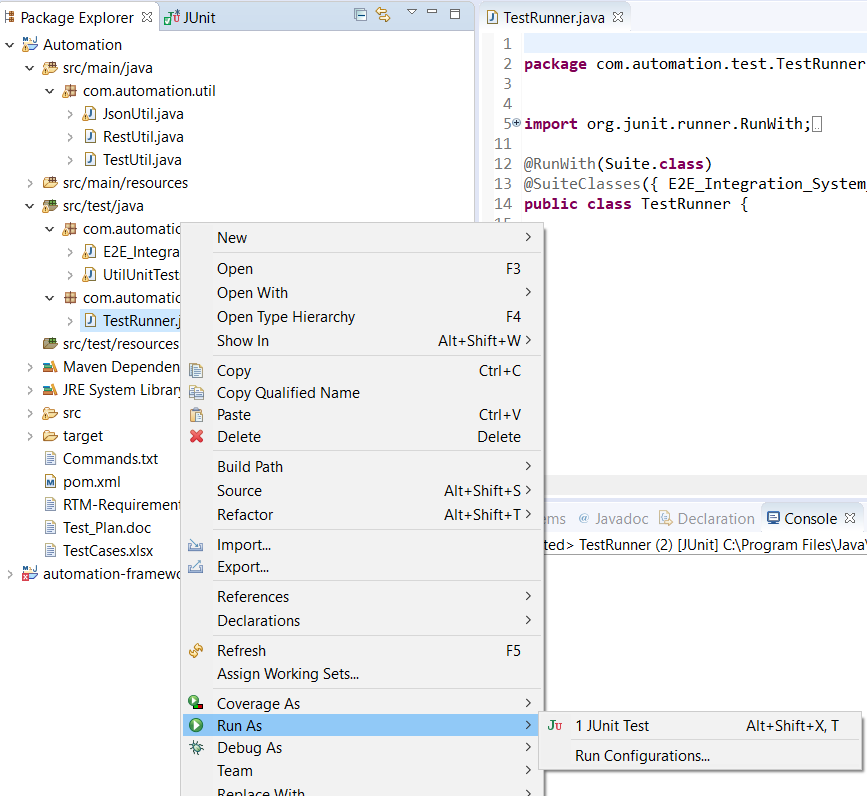
Step-1 create new folder and Pull the git code using below commands

$ git init

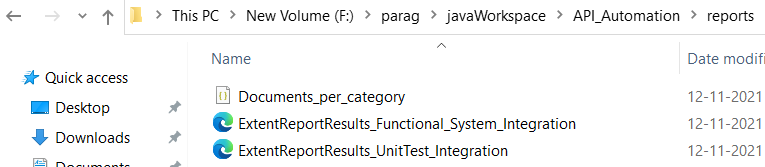
$ git pull <https://github.com/paragpatil7/API_Automation.git>

Step-2 Open Eclipse and import the above code

Step-3 Execute the TestRunner.java as Junit Test



Step-4 It will create the result in reports folder with Final Json of Documents per category and path.



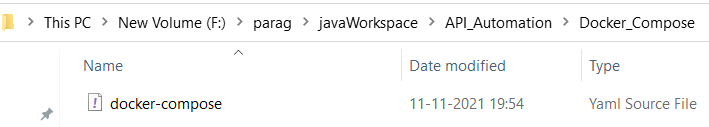
1. **Using Docker**

prerequisites

1- Please install the Docker and start the Docker

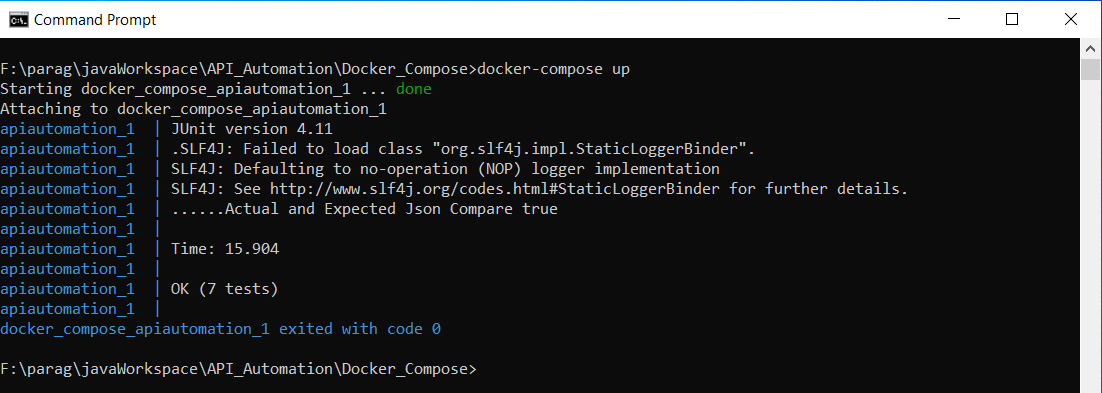
Step -1 Create new folder and Pull or download only Docker-Compose.yaml file from Git (no need of other code)

Git-Location - <https://github.com/paragpatil7/API_Automation/blob/main/Docker_Compose/docker-compose.yaml>

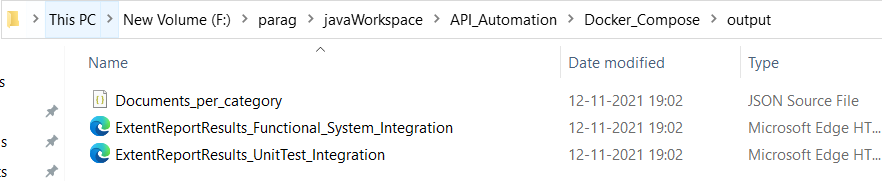


Step -2 Open cmd and go to the location where docker-compose.ymal file is stored and run command

docker-compose up



Also it will create the folder output with results and Json of Documents per category and path.



**Important Points**

1. **Code Coverage - EclEmma**

EclEmma is a free Java code coverage tool for [Eclipse](http://www.eclipse.org/), It brings code coverage analysis directly into the Eclipse workbench. EclEmma is based on the [JaCoCo](http://www.jacoco.org/jacoco) code coverage library.

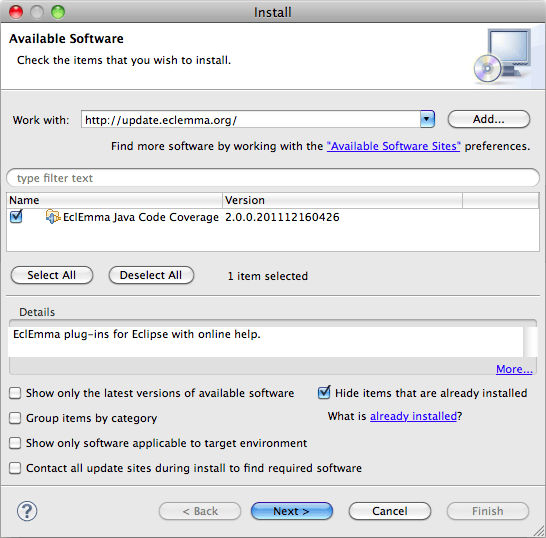
Installation

1. From your Eclipse menu select Help → Install New Software...

2. In the Install dialog enter http://update.eclemma.org/ at the Work with field.

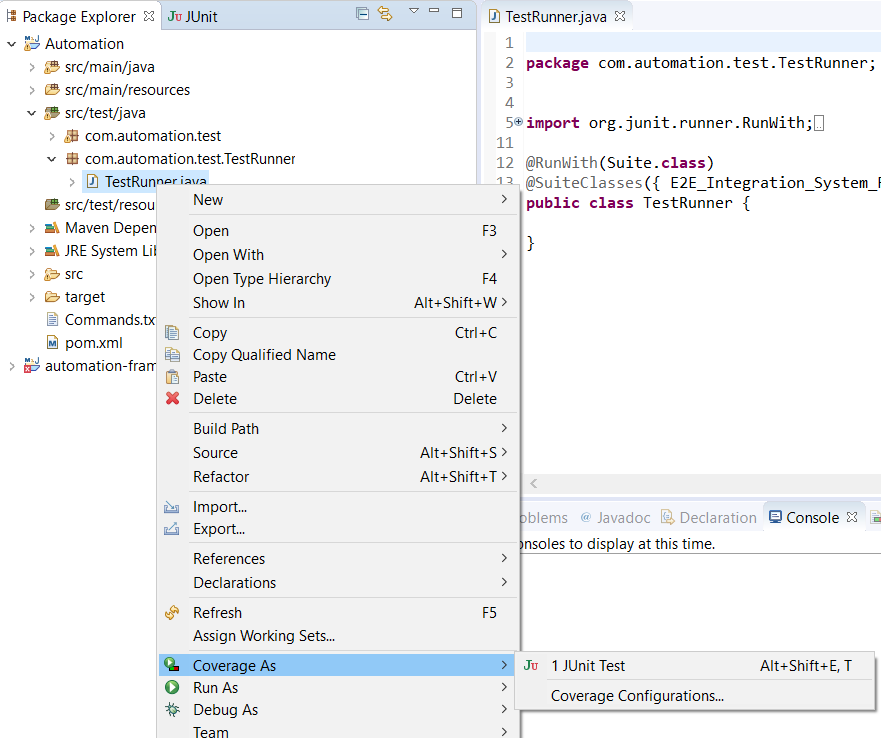
3. Check the latest EclEmma version and press Next

4. Follow the steps in the installation wizard.

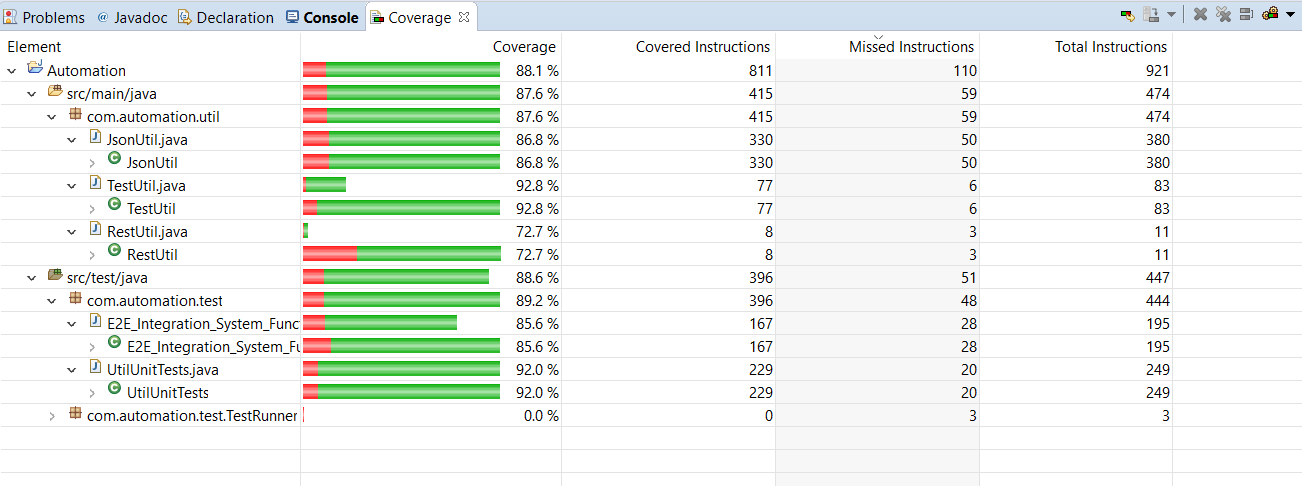


Forexecutionfrom eclipse go to Test Runner, right click on it select Coverage As Junit Test.

Then it will execute the all the Tests and it will show code coverage for that execution.

****

Code Coverage Report will look like below,

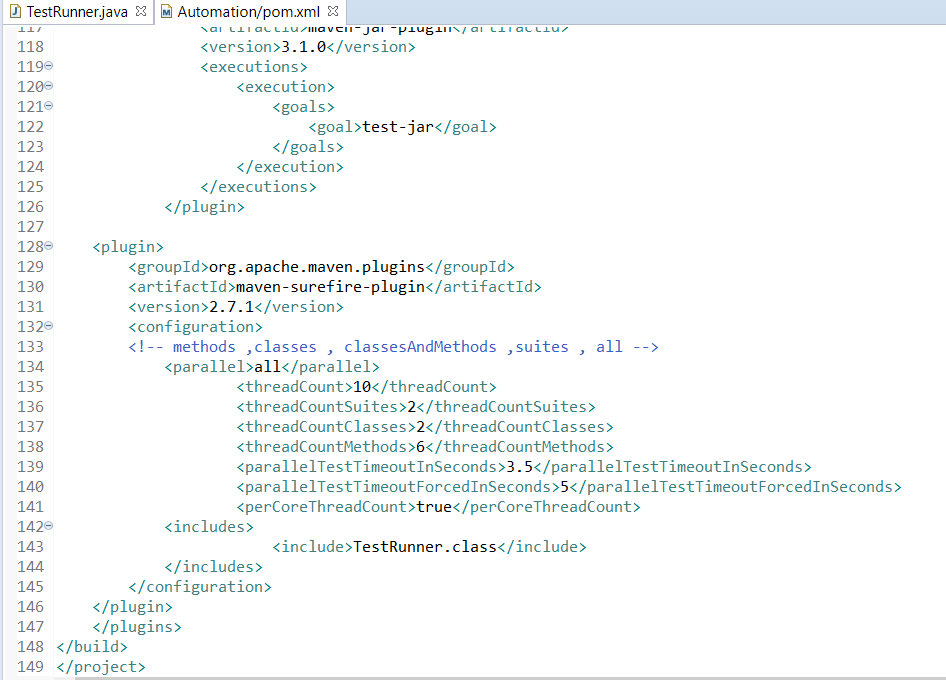
****

It will show the current execution code coverage % for each class.

1. **Parallel Execution**

Parallel testing using JUnit and Maven's Surefire Plugin

In POM.xml add plugin of Surefire Plugin and provide Parallel execution options and execution test as junit



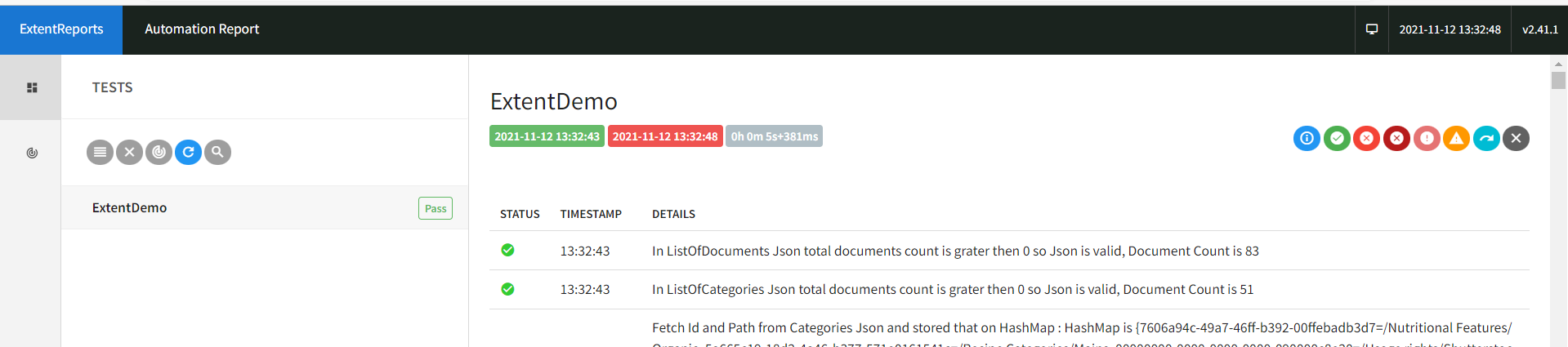
Provide options how you want parallel execution e.g.

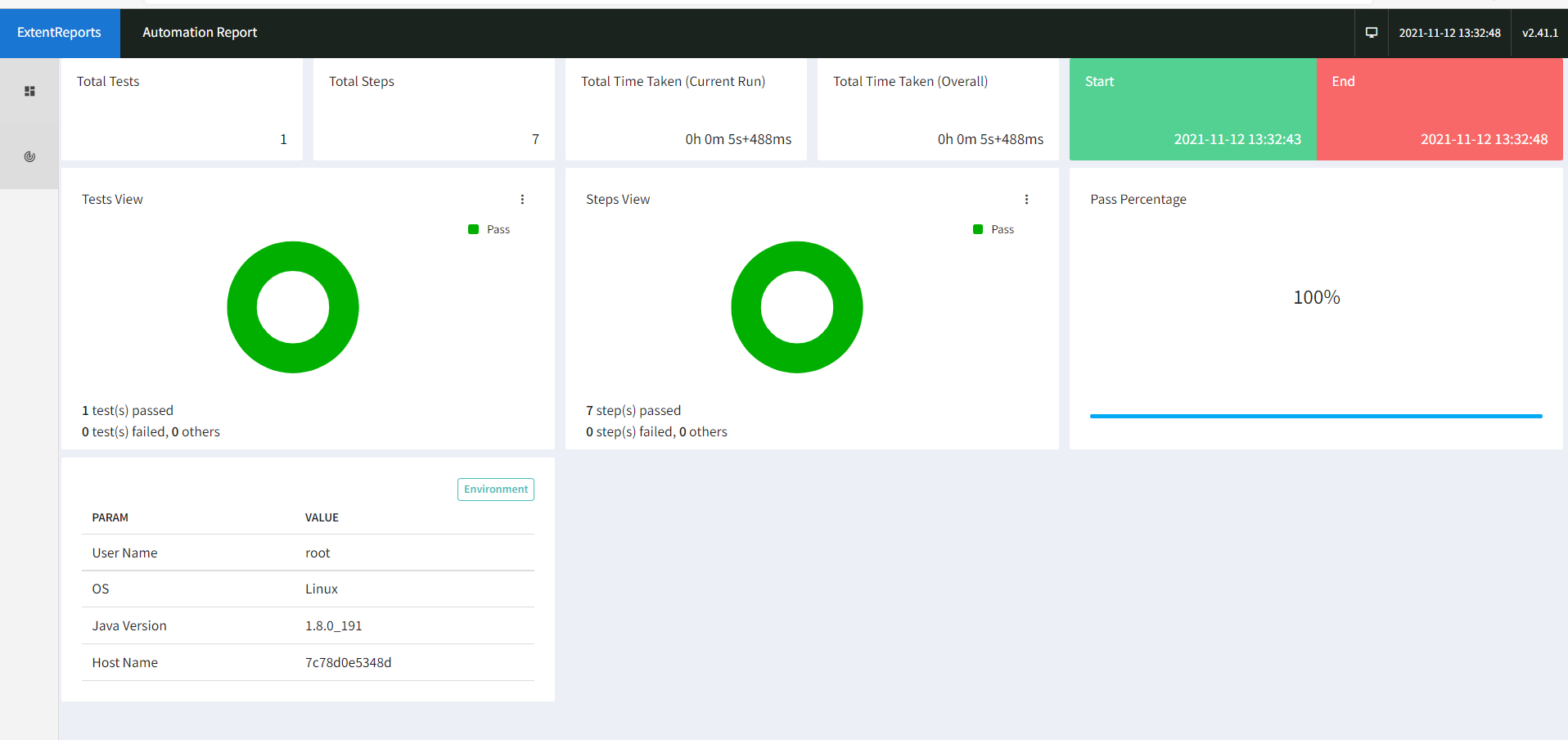
Parallel option – methods, Classes, classesAndMethods , Suits , all

If we provide option as Classes as parallel, then it will execute classes parallel.

3 – **Reports of the execution** will look like below

We have used **Extend Reports** Jar to create the Html reports with execution start and end time, total execution time, test status with Graph and steps details, no of steps, execution done by user name, OS name, Java version, host name.





Note:-

**For Any Issue or need any clarifications please contact me on below**

[**Paragpatil7@gmail.com**](mailto:Paragpatil7@gmail.com)

**+91 9975566880**