

What Is A Jenkins Job?

In simple words, any automated process that is implemented in Jenkins is a Jenkins Job. The automated process can be about building the source code. The source code can be merged from any of the source code management like git, SVN, and perforce.

Types Of Jenkins Jobs

Jenkins supports the following different types of job:

- #1) Run Jenkins as a standalone application
- #2) Log in to the Jenkins.
- #3) Click on “New Item” on the Jenkins dashboard as shown in Figure 1 below.

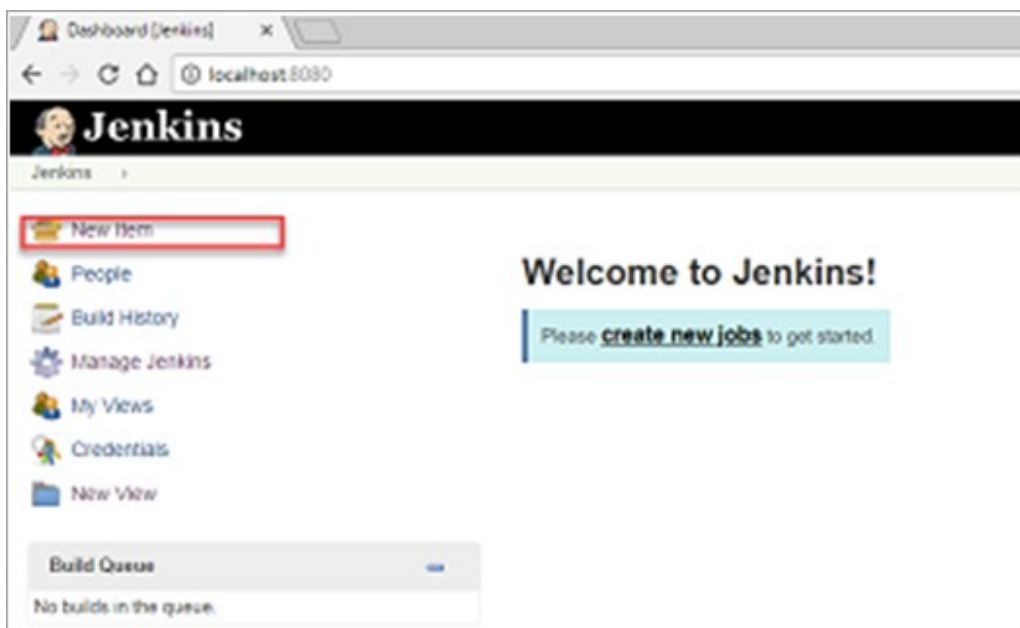


Figure 1: New Item in the Jenkins Dashboard

- #4) Upon clicking on a New Item, it will give the list of different Jenkins jobs. This is as shown in Figure 2.

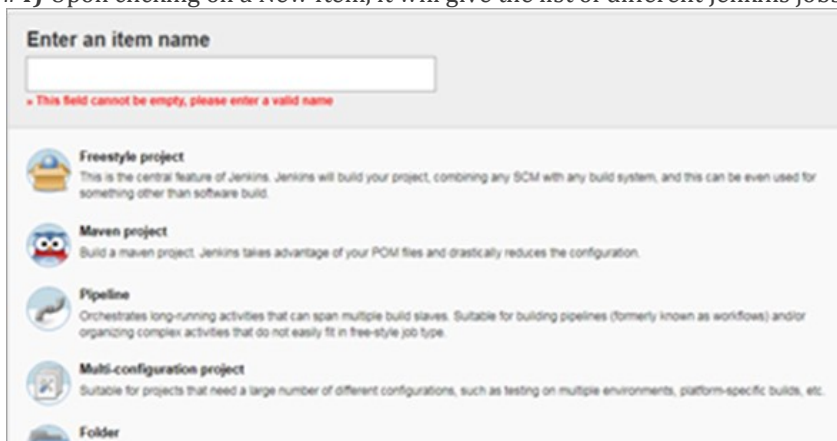


Figure 2: List of Jenkins Jobs

As shown in the above figure, different types of Jenkins Jobs are:

- (i) **Freestyle Project:** This is a regular and popular job in Jenkins which allows us to build our project, integrate our builds or source code management with Jenkins, poll the SCM, create triggers, and many more.
- (ii) **Maven Project:** Enables us to build our maven projects. We need to only specify the location of our pom.xml file to build the project. All other features like creating triggers, poll SCM remains the same.
- (iii) **Pipeline:** It is a kind of job which lets us declare the build process like compile, run, and report generations if required.

All the above steps can be mentioned in the file called Jenkinsfile in our code base and specify the same path in Jenkins as well. This will run the Jenkinsfile and show the stages of deployment like build, run, etc. This can be followed if we need to run on only one branch.

If we want to run our pipeline on multiple branches or versions of the codebase we make use of **the multi-branch pipeline**.

- (iv) **Multi-configuration:** This kind of project is for a large codebase that needs to be run on different configurations of operating systems.
- (v) **Folder:** In this kind of project, a folder which is a container for all other kinds of projects is created first. We can also make these folders secure.

Configuring Source Code Management

Following are pre-requisites for configuring source code management:

- **Git:** It's a version control tool used here. Any other source code management can be used as well like SVN, Perforce, and many more. When the Docker toolbox is loaded, even Git is loaded.
- **Code:** Code can be written easily using software like Eclipse, Microsoft Visual Studio, and many more.
- Jenkins up and running and launched using <http://localhost:8080>.

Following are the steps to configure Source Code Management:

#1) First, log in to the GitHub repository and select the repository created. I have already created a repository that contains a Java file having a simple selenium script and the print statement. Please refer to Figure 3.

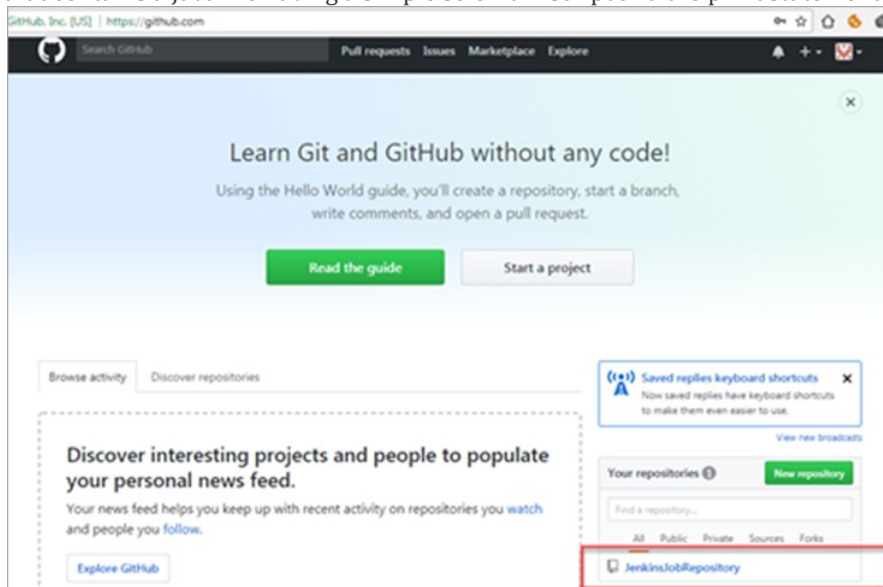


Figure 3: Logged in the screen of GitHub.

#2) Click on the repository and make sure that the latest code is present. Note the URL of the repository as shown in Figure 4.

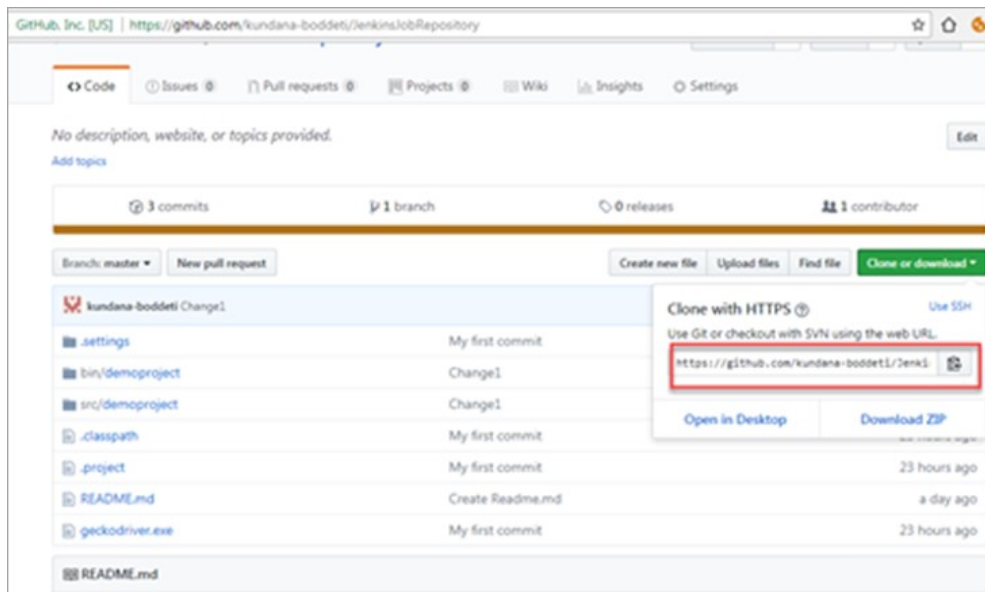


Figure 4: Screen showing GitHub repository and URL

#3) Now in Jenkins dashboard click on “New Item”. Refer Figure 1 for the Jenkins dashboard.

#4) Type a project name and select Freestyle project from a list of jobs enlisted as shown in Figure 5.

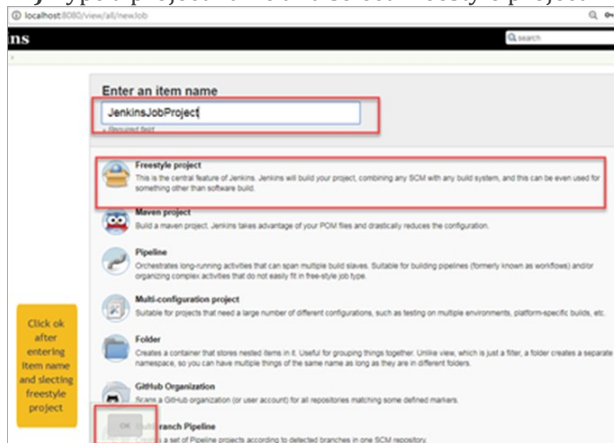


Figure 5: Creation of Freestyle project

#5) Select Git under source code management and provide the Github repository URL (already noted in step #2) and provide the credentials as shown in Figure 6 below.

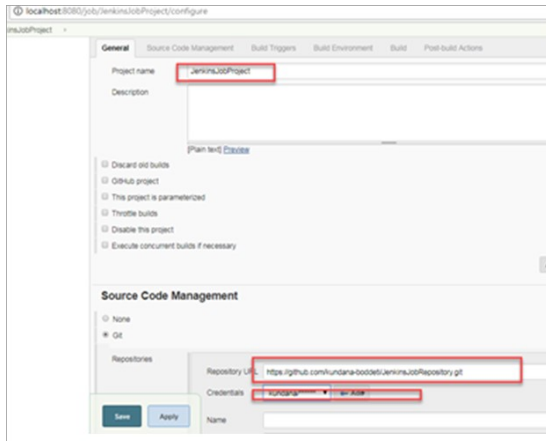


Figure 6: Freestyle project configuration

#6) Provide the required build commands to run the code. Here we have used execute windows batch command as shown in Figure 7 below.

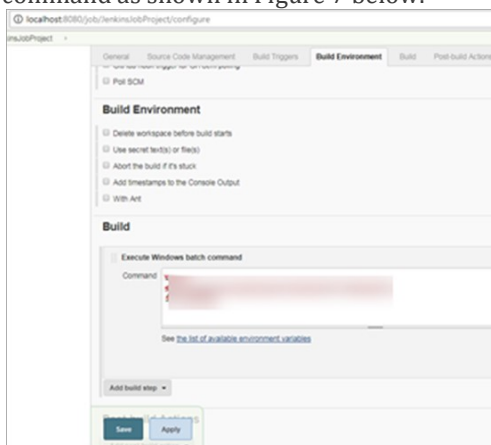


Figure 7: Build step in SCM configuration

#7) Click on Apply and Save button.

#8) Click on Build Now as shown in Figure 8.

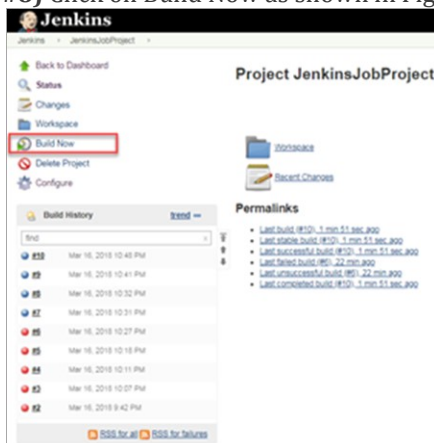


Figure 8: Build Now in a dashboard

#9) Find the result in the console output as shown in Figure 9.

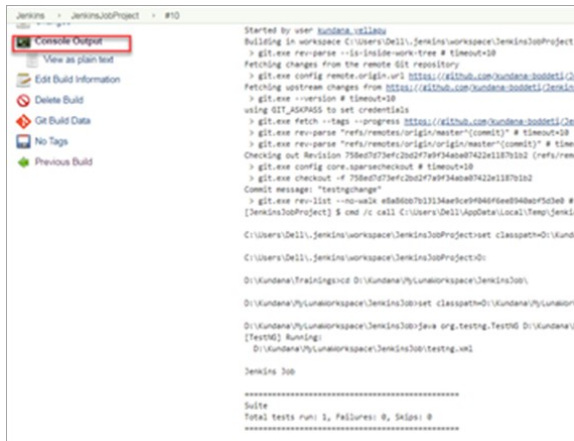


Figure 9: Console Output under the project

Note: To get the console output, click on the build result (indicated in blue or red). This will take you to the screen shown in Figure 9.

Build Triggers

What is a trigger?

A trigger lets us execute a job on an event occurrence. This event is called a trigger. To see the list of build triggers, we need to login to Jenkins and click on any item already created and click on configure.

The list of triggers is shown in Figure 10 below:

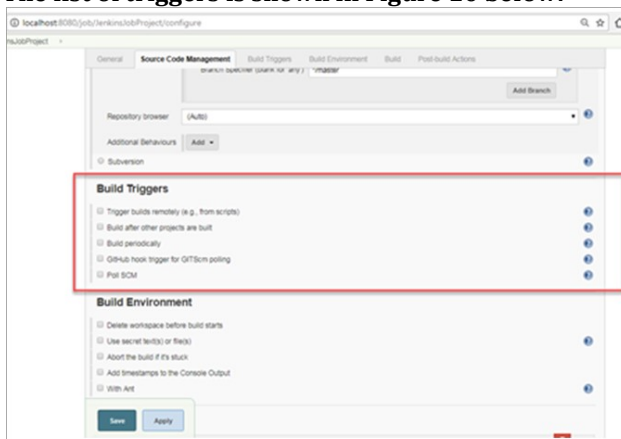


Figure 10: List of build triggers

- **Trigger build remotely:** The job is usually triggered by accessing a specified URL. This is convenient for scripts. With the URL, one needs to mention the authorization token as well.
- **Build after other projects are built:** As it reads, we need to mention the list of other projects, once those projects are built then the present job is executed.
- **Build periodically:** The build is triggered based on the mentioned time. A cron has to be mentioned here.
- **Github hook trigger for GITSCM polling:** If Jenkins receives push GitHub hook from a repository associated with git, then the build process gets executed.
- **Poll SCM:** Configure Jenkins to poll the SCM for ant pushes or commits and then trigger the jobs.

Jenkins Job Scheduler

We need to do the following to schedule a job to build :

1. Log in to the dashboard of Jenkins.
2. Click on an item or job.
3. Click on Configure.

4. Check the build periodically option and set the desired cron as shown in Figure 11.

Note: Only for demo purpose, we are giving cron as `*****` which means trigger the build every minute. Please disable it or give a cron of your choice.

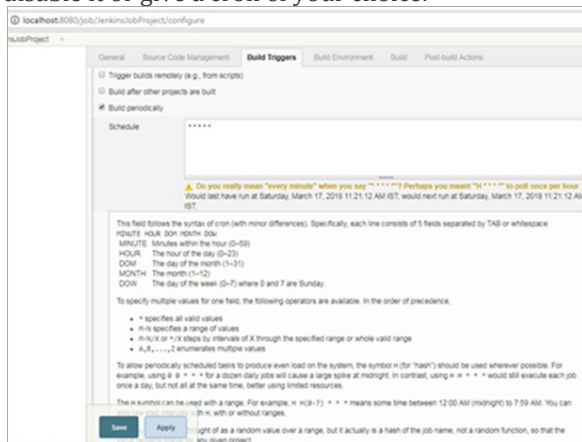


Figure 11: Schedule the build jobs

Polling the SCM

Follow the below steps:

1. Click on the "Configure" of the job created in the Jenkins dashboard.
2. Click on build triggers in the configure settings and select the Poll SCM.
3. Enter the desired cron to poll the SCM. Here we have given `* * * *` which means the Jenkins polls the SCM every minute. Please refer to Figure 12.

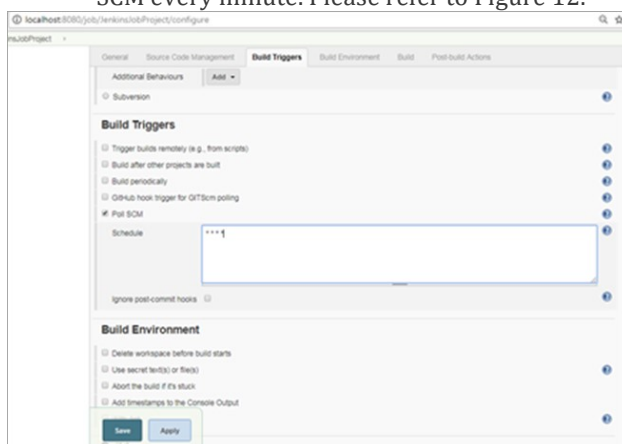


Figure 12: Poll the SCM

Maven Build Steps

- #1) Run the Jenkins and login to it.
- #2) Click on Manage Jenkins
- #3) Click on Global Tool Configuration.
- #4) Under Maven, click on add maven.
- #5) Uncheck install automatically.
- #6) Provide the path for the Maven bin as shown in Figure 13.

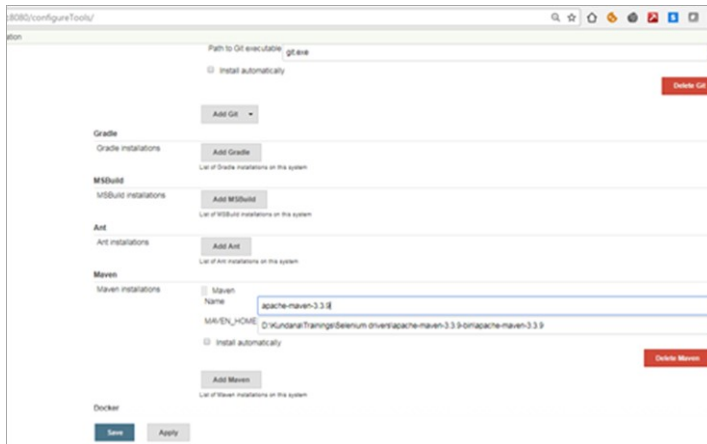


Figure 13: Adding the Maven path

#7) Click on apply and save.

#8) Go to Jenkins dashboard, click on New Item.

#9) Enter Item name and Select Maven project and click on Apply as shown in Figure 14.

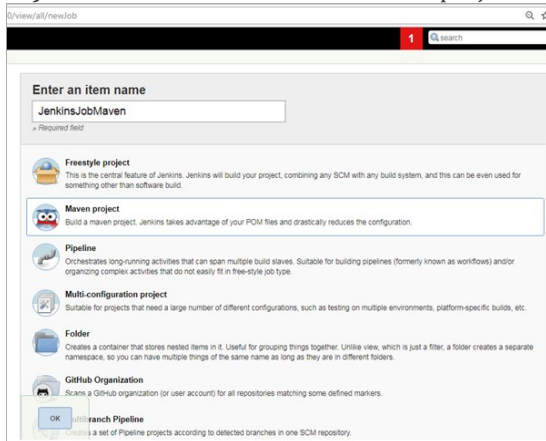


Figure 14: Creation of the Maven Project

#10) In the configure page of the Maven, enter a description, and under Maven click on advanced.

#11) Choose a custom workspace as shown in Figure 15. Custom workspace is the path where pom.xml is present.

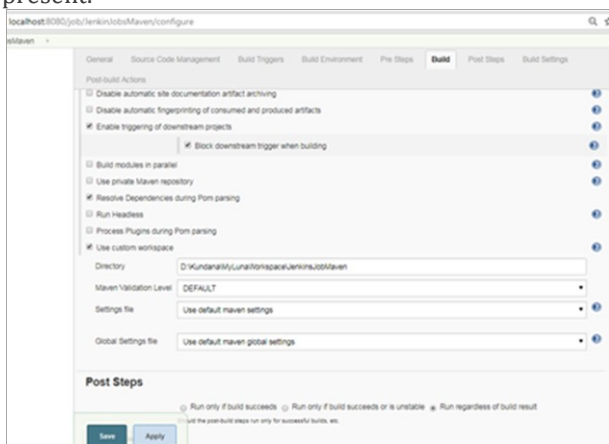


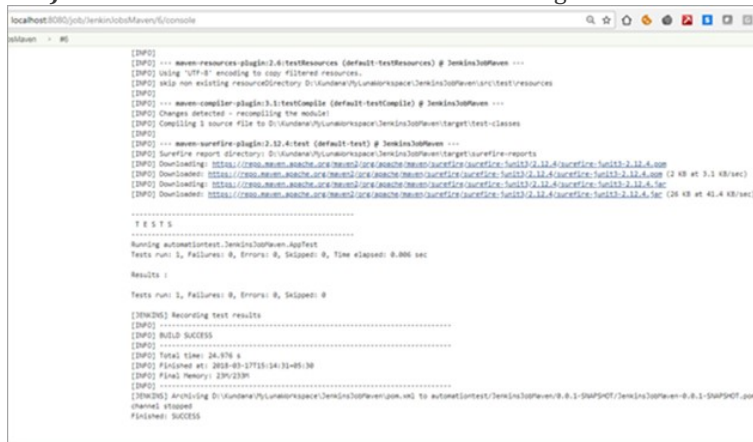
Figure 15: Custom Workspace under maven

#12) Under build, set the goals and options to clean compile test.

#13) Click on Save.

#14) Click on Build Now which is present in New Item.

#15) Wait till the build is a success as shown in Figure 16.



```
localhost:8080/job/JenkinsJobMaven/console
Jenkins - #5

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ JenkinsJobMaven ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\Jenkins\workspace\JenkinsJobMaven\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ JenkinsJobMaven ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to D:\Jenkins\workspace\JenkinsJobMaven\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ JenkinsJobMaven ---
[INFO] Surefire report directory: D:\Jenkins\workspace\JenkinsJobMaven\target\surefire-reports
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-2.12.4.jar (2 KB at 5.3 KB/sec)
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-2.12.4-uber.jar (28 KB at 41.4 KB/sec)
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-2.12.4-uber.jar (28 KB at 41.4 KB/sec)

-----
T E S T S
-----
Running automatestest.JenkinsJobMaven.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.006 sec

Results:
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] Recording test results
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 24.576 s
[INFO] Finished at: 2018-03-17T15:14:13+05:30
[INFO] Final Memory: 23M/23M
[INFO] -----
[INFO] Archiving D:\Jenkins\workspace\JenkinsJobMaven\pom.xml to automatestest\JenkinsJobMaven\0.8.1-SNAPSHOT\JenkinsJobMaven-0.8.1-SNAPSHOT.pom
channel stopped
Finished: SUCCESS
```

Figure 16: Build Success

Conclusion

In this tutorial we learned about different types of Jenkins jobs, what are build triggers, configuring the SCM, polling the SCM, scheduling a job, and finally creating a Maven project.