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: Please refer to the tutorial "Installing and Running Jenkins" to get the detailed steps.
#2) Log in to the Jenkins.
#3) Click on "New Item" on the Jenkins dashboard as shown in Figure 1 below.

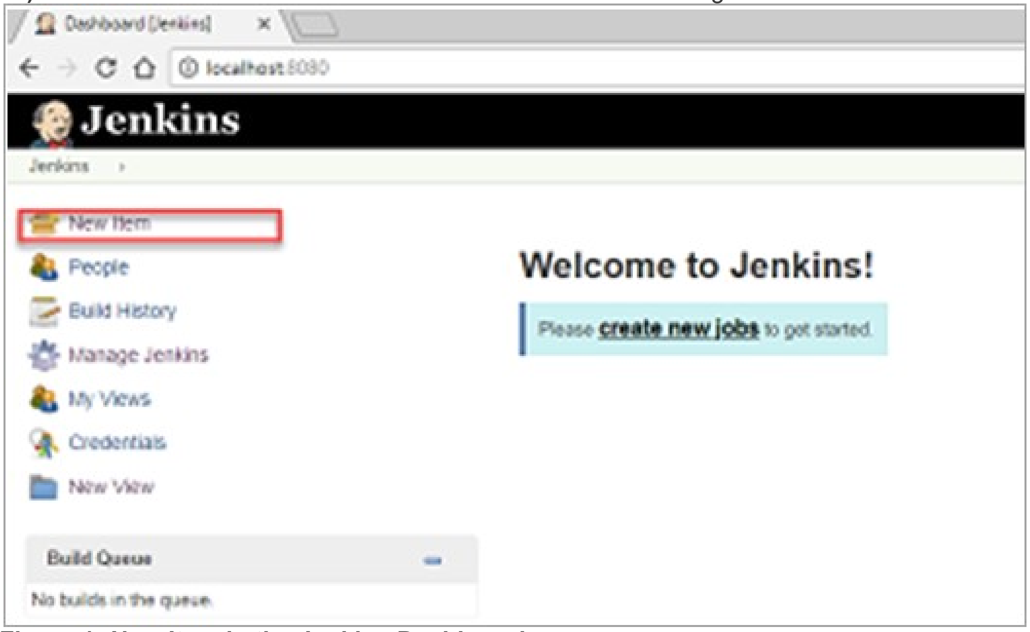
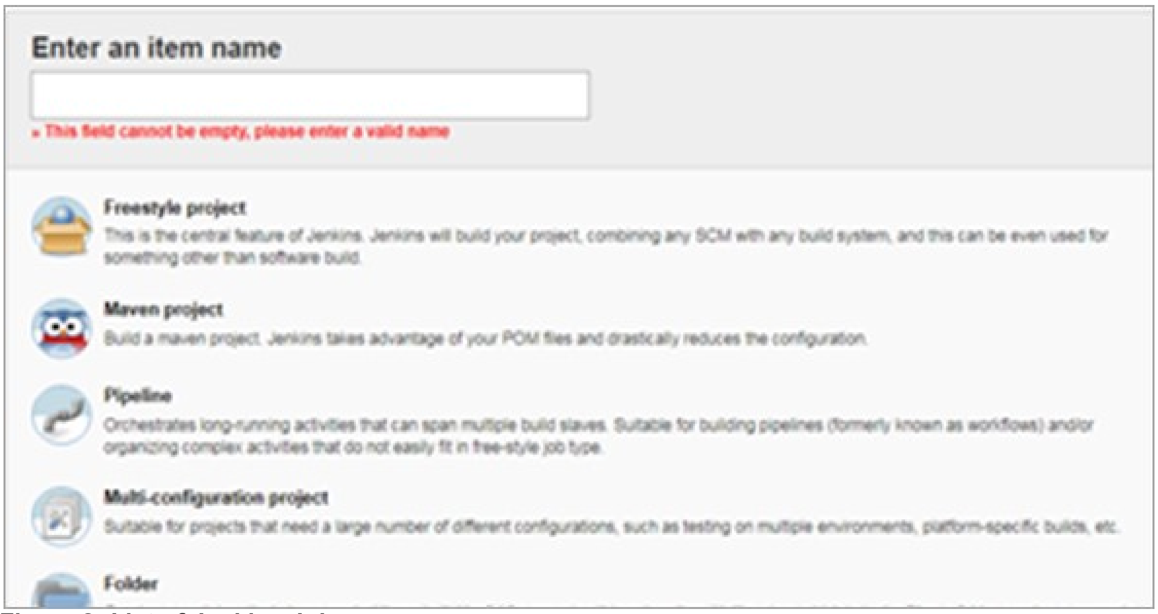


Figure 1: New Item in the Jenkins Dashboard
#4) Upon clicking on a New Item, it will give the list of different Jenkins jobs. This is as shown in Figure 2.


Enter an item name

» This field cannot be empty, please enter a valid name

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder

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 multibranch 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. The installation steps of the Docker toolbox are mentioned in my tutorial "Installing and Running Jenkins".
- **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 Mgement:

#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.

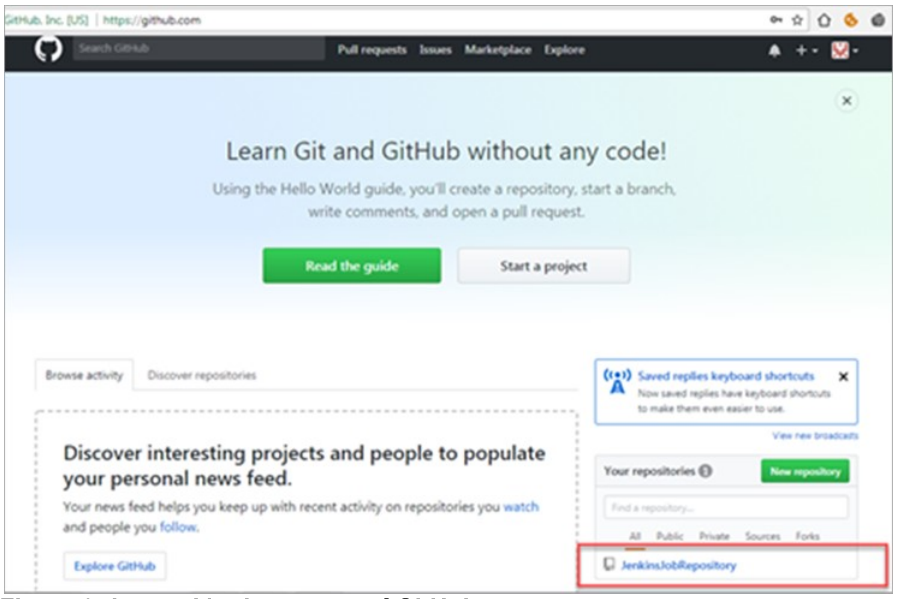

Learn Git and GitHub without any code!
Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.
Read the guide
Start a project
Discover interesting projects and people to populate your personal news feed.
Your repositories
New repository
JenkinsJobRepository

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.

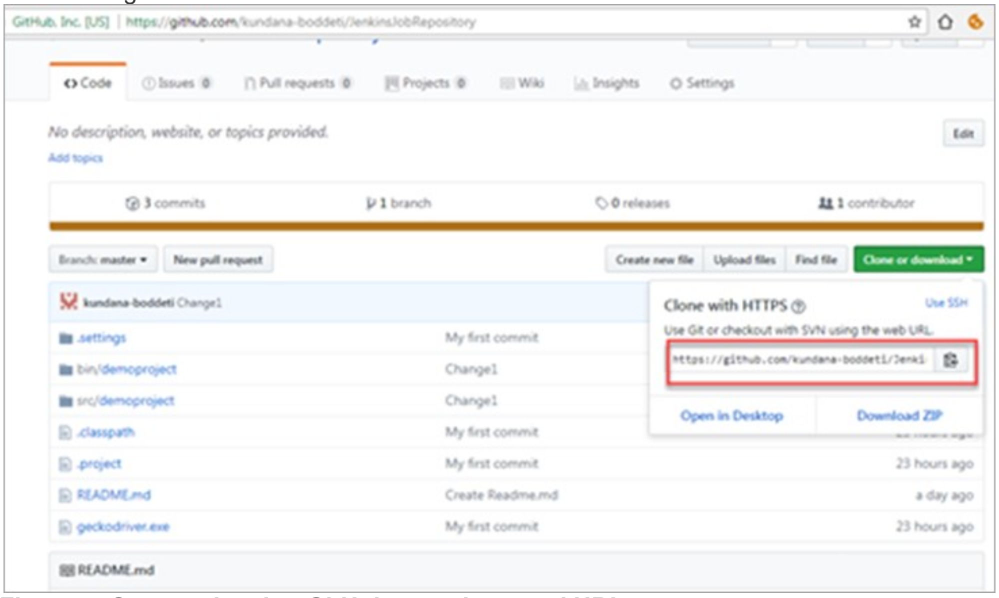
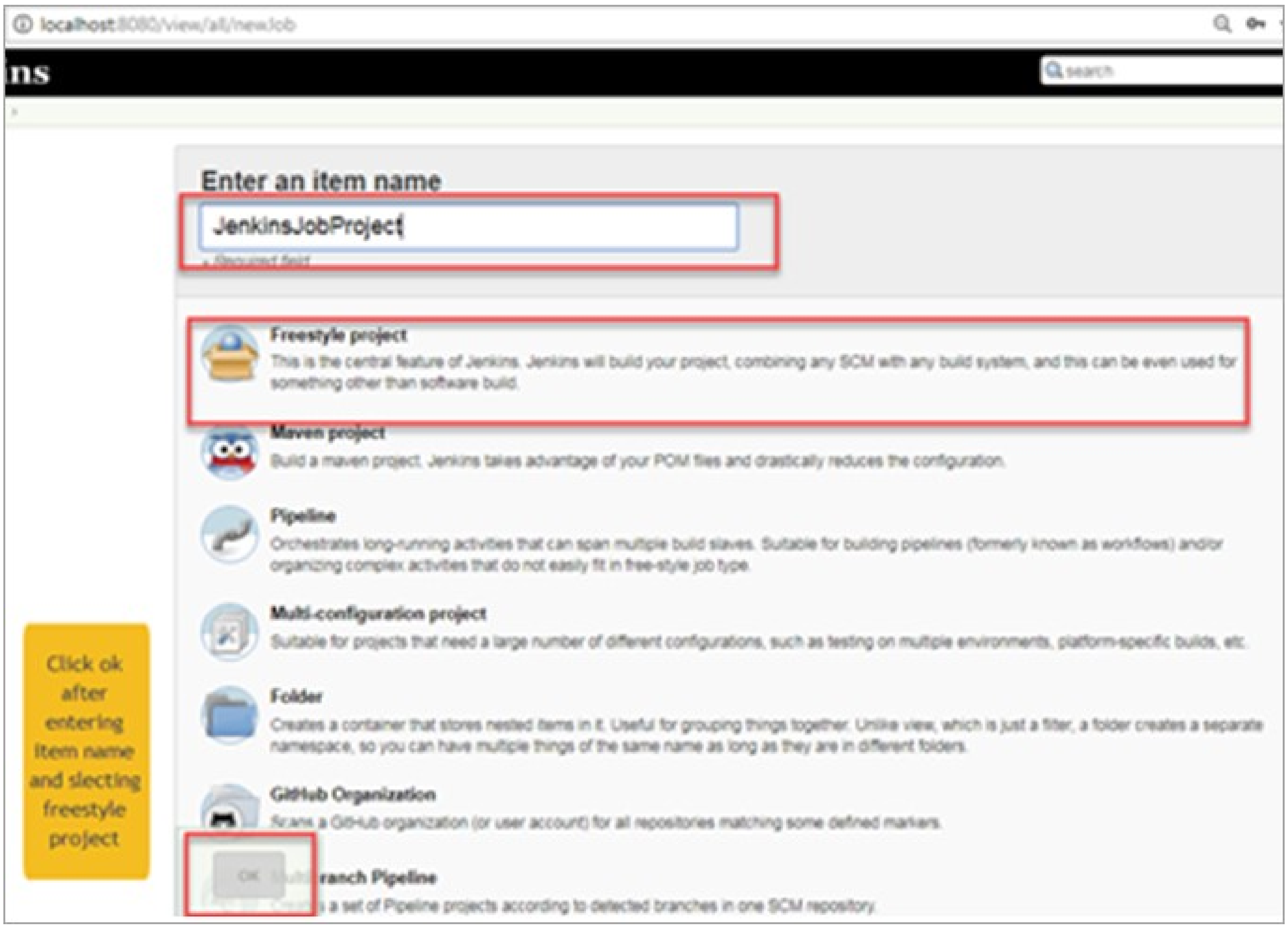


Figure 4: Screen showing GitHub repository and URL

#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.


localhost:8080/view/all/newJob

Enter an item name

JenkinsJobProject

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

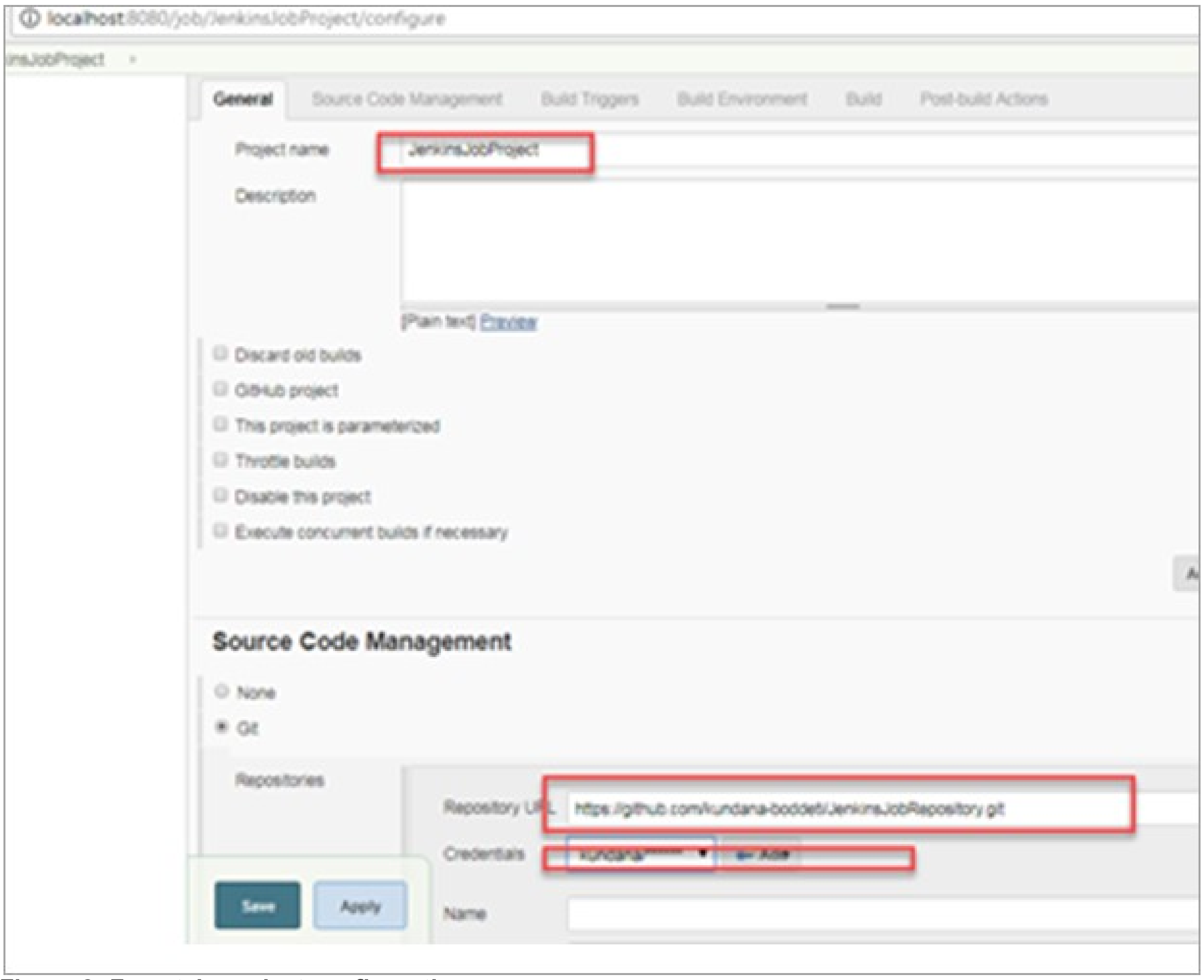
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Click ok after entering Item name and slecting freestyle project

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.


localhost:8080/job/JenkinsJobProject/configure

Project name: JenkinsJobProject

Description:

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary

Source Code Management

- ( ) None
- (•) Git

Repositories

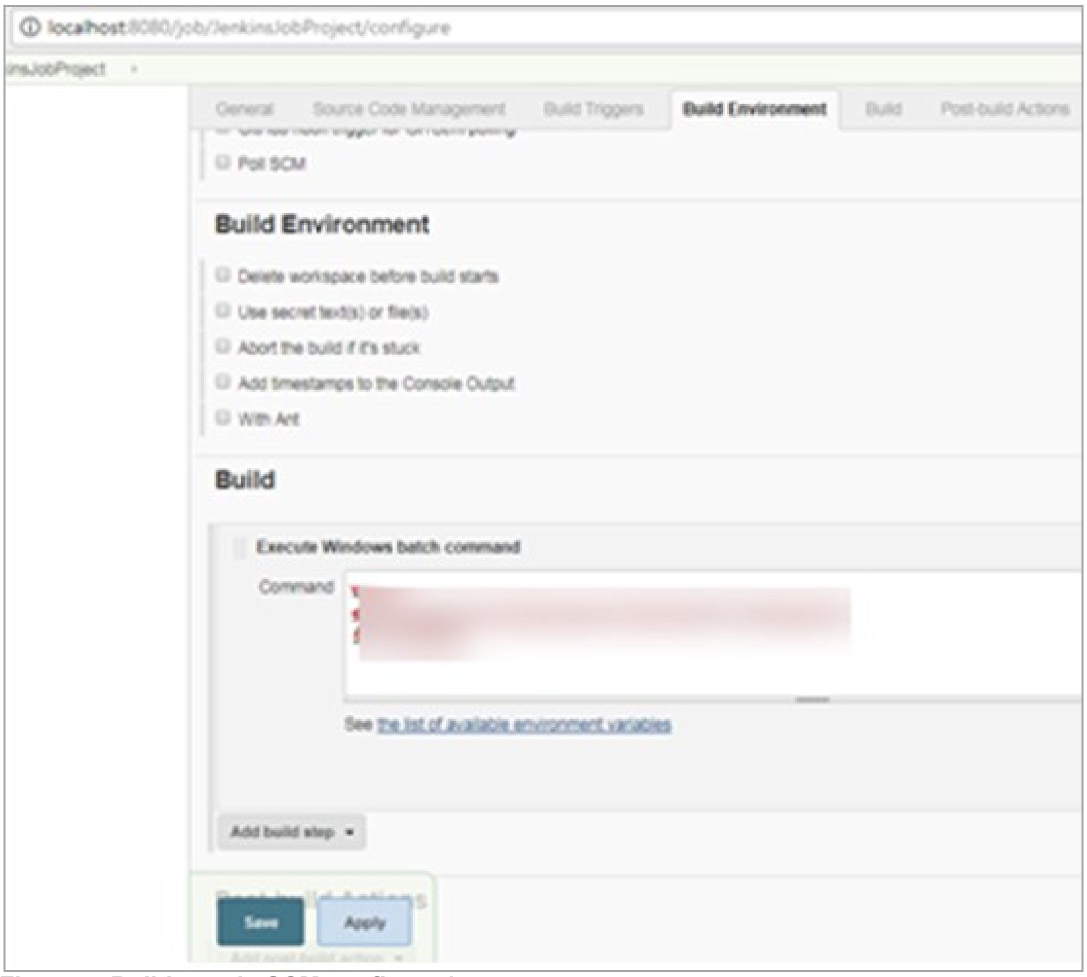
Repository URL: https://github.com/kundana-boddeti/JenkinsJobRepository.git

Credentials: kundana/******

Name:

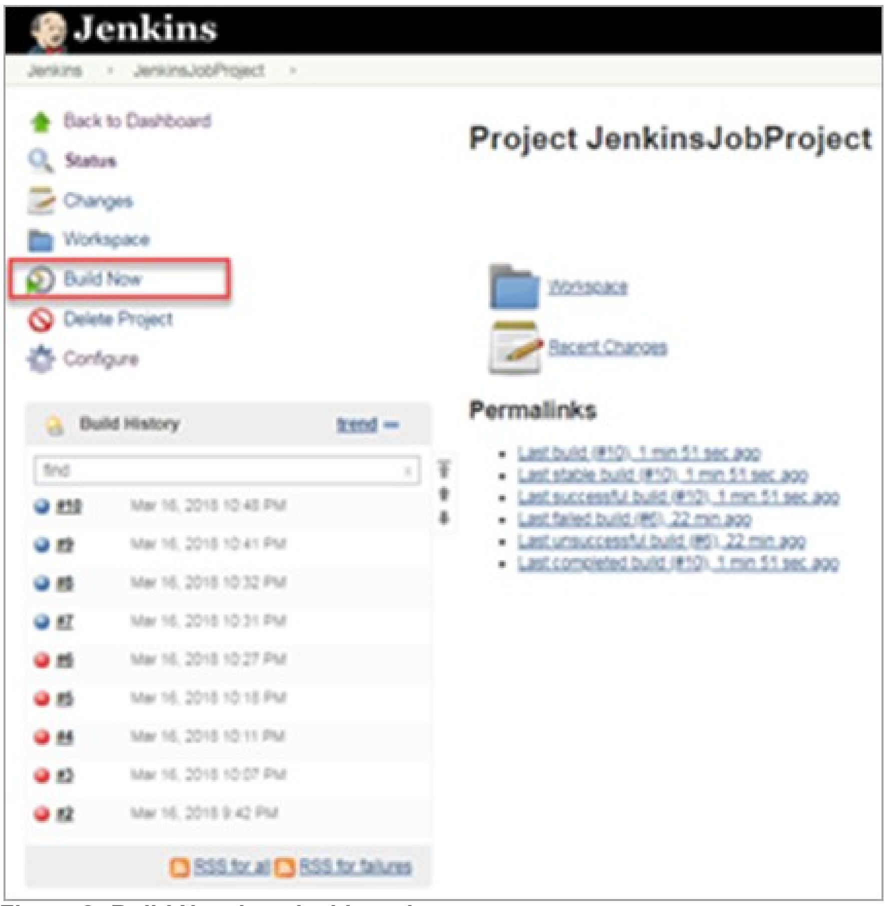
Figure 6: Freestyle project configuration

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


localhost:8080/job/JenkinsJobProject/configure
insJobProject
General
Source Code Management
Build Triggers
Build Environment
Build
Post-build Actions
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Abort the build if it's stuck
Add timestamps to the Console Output
With Ant
Build
Execute Windows batch command
Command
See the list of available environment variables
Add build step
Save
Apply

[redacted]

Figure 7: Build step in SCM configuration
#7) Click on Apply and Save button.
#8) Click on Build Now as shown in Figure 8.


Jenkins

Jenkins › JenkinsJobProject ›

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure

Build History trend

#10	Mar 16, 2018 10:48 PM
#9	Mar 16, 2018 10:41 PM
#8	Mar 16, 2018 10:32 PM
#7	Mar 16, 2018 10:31 PM
#6	Mar 16, 2018 10:27 PM
#5	Mar 16, 2018 10:18 PM
#4	Mar 16, 2018 10:11 PM
#3	Mar 16, 2018 10:07 PM
#2	Mar 16, 2018 9:42 PM

RSS for all RSS for failures

Project JenkinsJobProject

Workspace

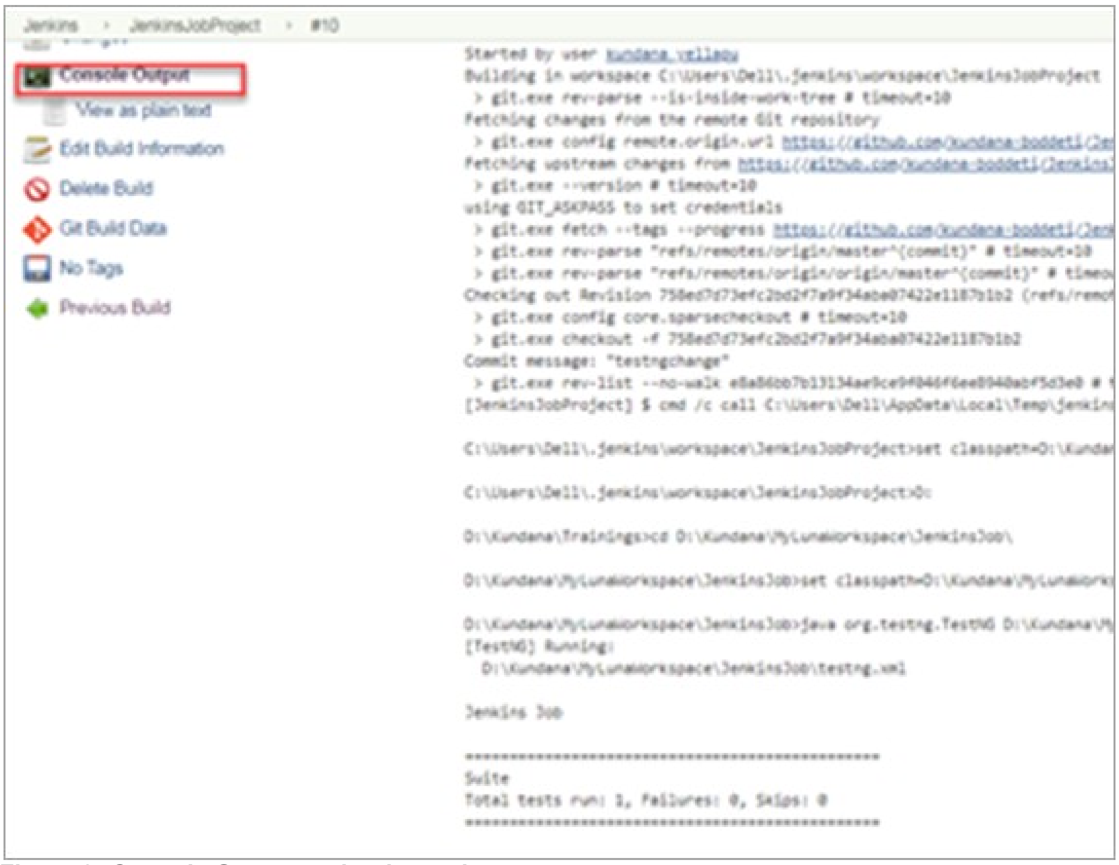
Recent Changes

Permalinks

- Last build (#10), 1 min 51 sec ago
- Last stable build (#10), 1 min 51 sec ago
- Last successful build (#10), 1 min 51 sec ago
- Last failed build (#6), 22 min ago
- Last unsuccessful build (#6), 22 min ago
- Last completed build (#10), 1 min 51 sec ago

Figure 8: Build Now in a dashboard

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



```
Started by user kundana.vellanu
Building in workspace C:\Users\Dell\.jenkins\workspace\JenkinsJobProject
 > git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git.exe config remote.origin.url https://github.com/kundana-boddeti/Jen
Fetching upstream changes from https://github.com/kundana-boddeti/Jenkins
 > git.exe --version # timeout=10
using GIT_ASKPASS to set credentials
 > git.exe fetch --tags --progress https://github.com/kundana-boddeti/Jenk
 > git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
 > git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeou
Checking out Revision 758ed7d73efc2bd2f7a9f34aba07422e1187b1b2 (refs/remot
 > git.exe config core.sparsecheckout # timeout=10
 > git.exe checkout -f 758ed7d73efc2bd2f7a9f34aba07422e1187b1b2
Commit message: "testngchange"
 > git.exe rev-list --no-walk e8a86bb7b13134ae9ce9f046f6ee8940abf5d3e0 # t
[JenkinsJobProject] $ cmd /c call C:\Users\Dell\AppData\Local\Temp\jenkin

C:\Users\Dell\.jenkins\workspace\JenkinsJobProject>set classpath=D:\Kunda

C:\Users\Dell\.jenkins\workspace\JenkinsJobProject>D:

D:\Kundana\Trainings>cd D:\Kundana\MyLunaWorkspace\JenkinsJob\

D:\Kundana\MyLunaWorkspace\JenkinsJob>set classpath=D:\Kundana\MyLunaWork

D:\Kundana\MyLunaWorkspace\JenkinsJob>java org.testng.TestNG D:\Kundana\My
[TestNG] Running:
  D:\Kundana\MyLunaWorkspace\JenkinsJob\testng.xml

Jenkins Job

===============================================
Suite
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

Figure 9: Console Output under the project

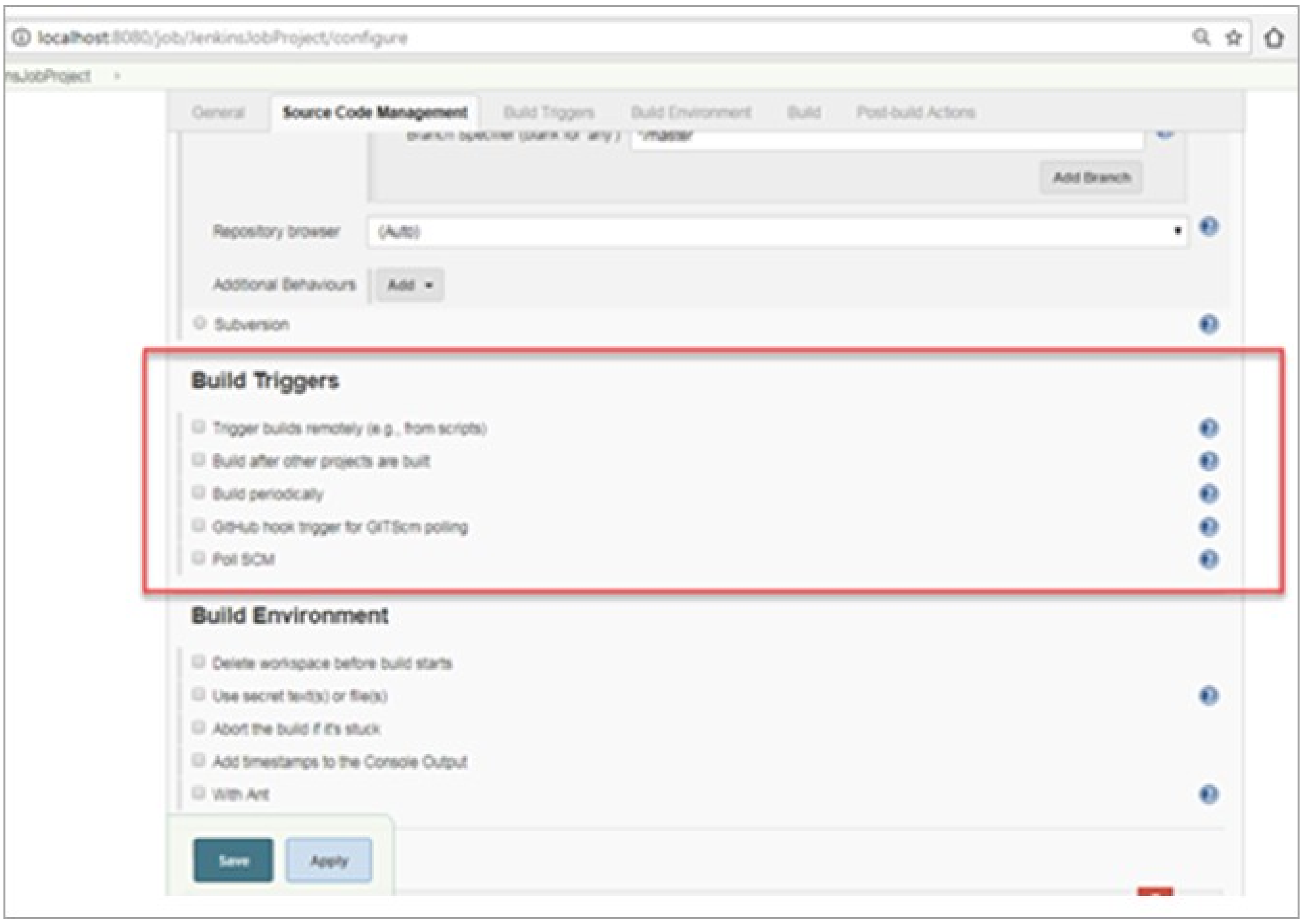
Note: To get the console output, click on the build result (indicated in blue or red). This will take you to the screen shown in Figure 9.

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:


localhost:8080/job/JenkinsJobProject/configure

nsJobProject ›

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Branch Specifier (blank for 'any') */master

Add Branch

Repository browser (Auto)

Additional Behaviours Add

Subversion

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Build Environment

- Delete workspace before build starts
- Use secret text(s) or file(s)
- Abort the build if it's stuck
- Add timestamps to the Console Output
- With Ant

Save | Apply

Figure 10: List of build triggers

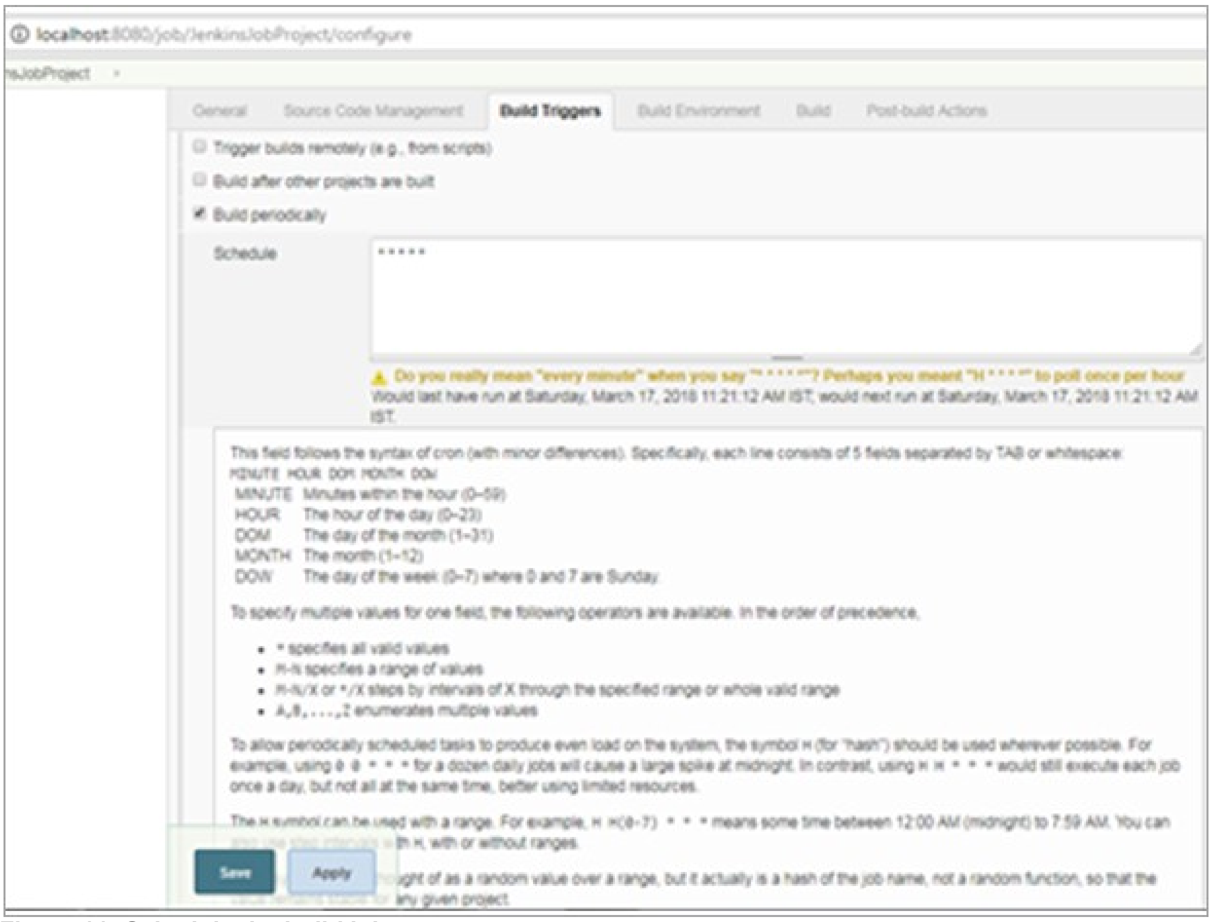
- **Trigger build remotely**: The job is usually triggered by accessing a specified URL. This is convenient for scripts. With the URL, one needs to mention the authorization token as well.
- **Build after other projects are built**: As it reads, we need to mention the list of other projects, once those 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.



☐ Trigger builds remotely (e.g., from scripts)

☐ Build after other projects are built

☑ Build periodically

Schedule: `* * * * *`

⚠ Do you really mean "every minute" when you say "* * * * *"? Perhaps you meant "H * * * *" to poll once per hour
Would last have run at Saturday, March 17, 2018 11:21:12 AM IST; would next run at Saturday, March 17, 2018 11:21:12 AM IST.

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

```
MINUTE HOUR DOM MONTH DOW
 MINUTE  Minutes within the hour (0–59)
 HOUR    The hour of the day (0–23)
 DOM     The day of the month (1–31)
 MONTH   The month (1–12)
 DOW     The day of the week (0–7) where 0 and 7 are Sunday.
```

To specify multiple values for one field, the following operators are available. In the order of precedence,

- `*` specifies all valid values
- `M-N` specifies a range of values
- `M-N/X` or `*/X` steps by intervals of X through the specified range or whole valid range
- `A,B,...,Z` enumerates multiple values

To allow periodically scheduled tasks to produce even load on the system, the symbol `H` (for "hash") should be used wherever possible. For example, using `0 0 * * *` for a dozen daily jobs will cause a large spike at midnight. In contrast, using `H H * * *` would still execute each job once a day, but not all at the same time, better using limited resources.

The `H` symbol can be used with a range. For example, `H H(0-7) * * *` means some time between 12:00 AM (midnight) to 7:59 AM. You can [illegible] with `H`, with or without ranges.

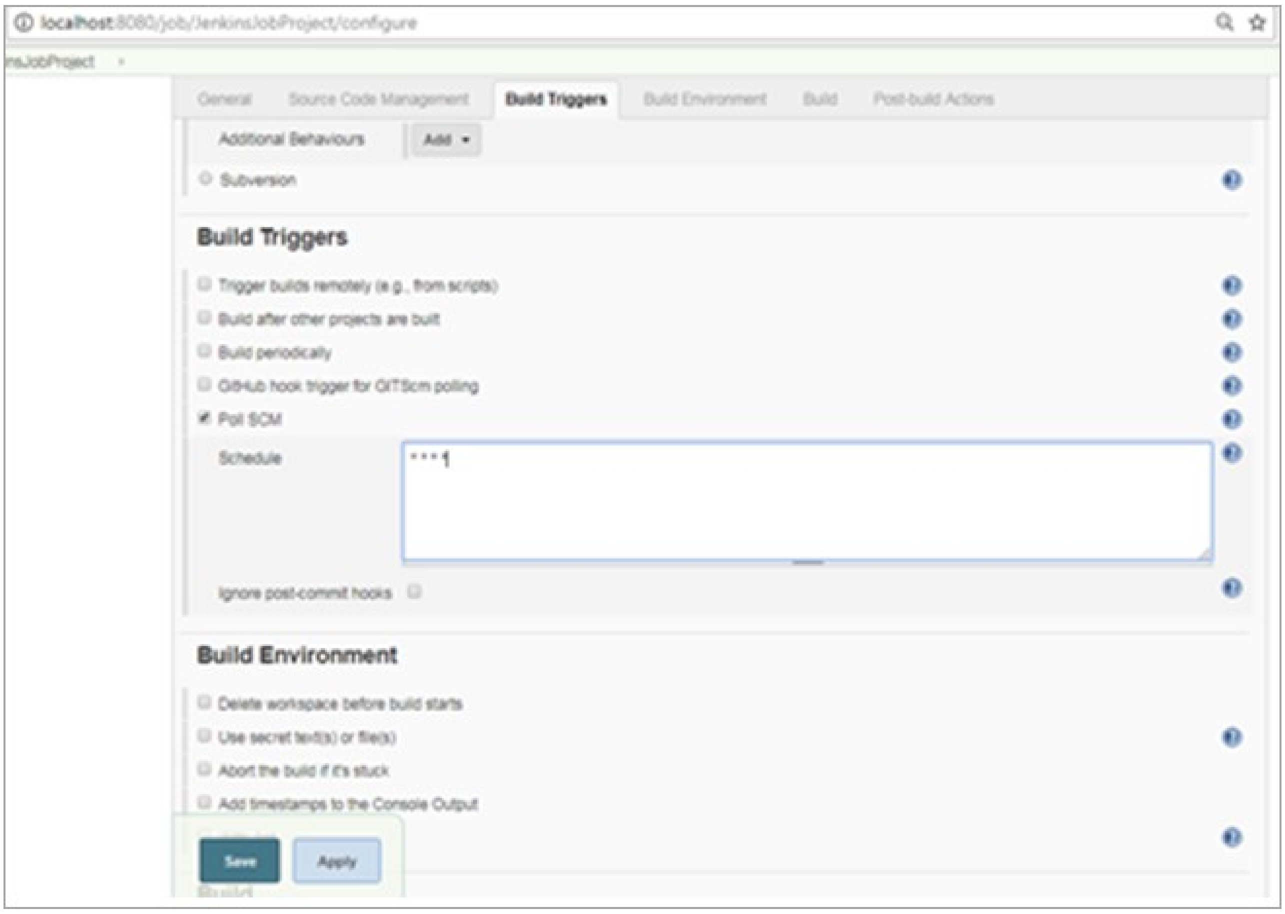
[illegible] ought of as a random value over a range, but it actually is a hash of the job name, not a random function, so that the [illegible] any given project.

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.


localhost:8080/job/JenkinsJobProject/configure

nsJobProject >

General | Source Code Management | Build Triggers | Build Environment | Build | Post-build Actions

Additional Behaviours [Add]

○ Subversion

Build Triggers

- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☑ Poll SCM

Schedule: `* * * *`

Ignore post-commit hooks ☐

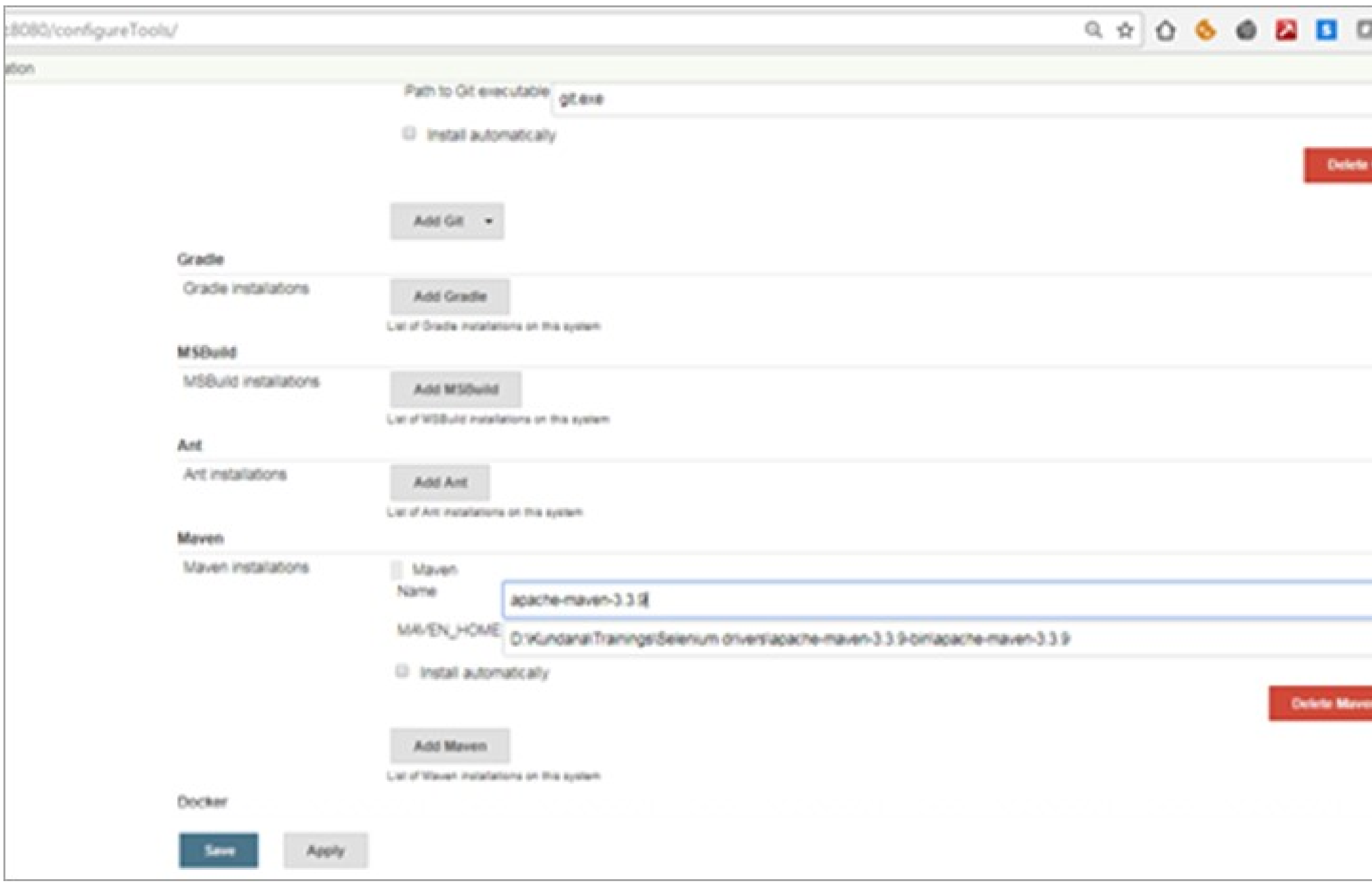
Build Environment

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s)
- ☐ Abort the build if it's stuck
- ☐ Add timestamps to the Console Output

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.


:8080/configureTools/

Path to Git executable: git.exe

Install automatically

Gradle

Gradle installations

List of Gradle installations on this system

MSBuild

MSBuild installations

List of MSBuild installations on this system

Ant

Ant installations

List of Ant installations on this system

Maven

Maven installations

Maven

Name: apache-maven-3.3.9

MAVEN_HOME: D:\Kundana\Trainings\Selenium drivers\apache-maven-3.3.9-bin\apache-maven-3.3.9

Install automatically

List of Maven installations on this system

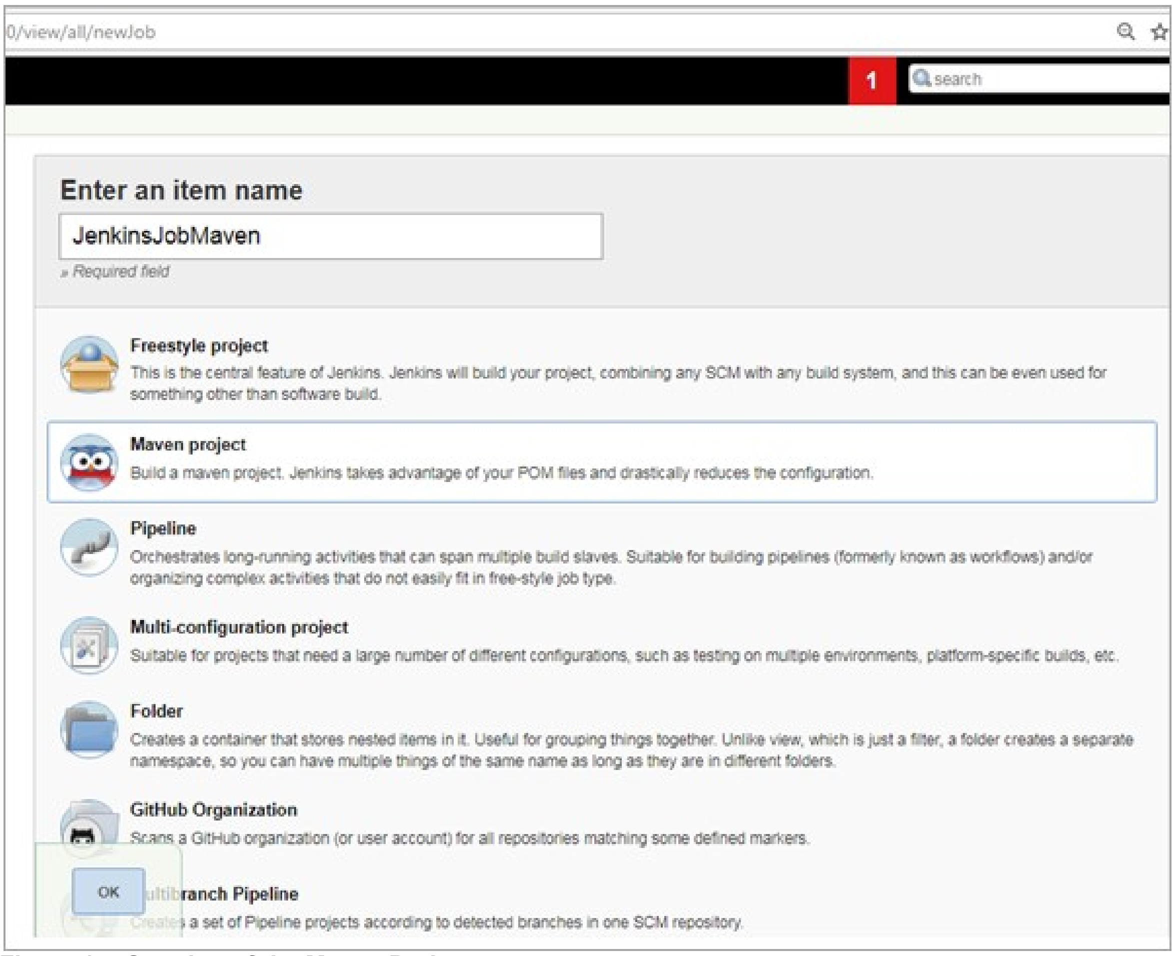
Docker

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.



0/view/all/newJob

Enter an item name

JenkinsJobMaven

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

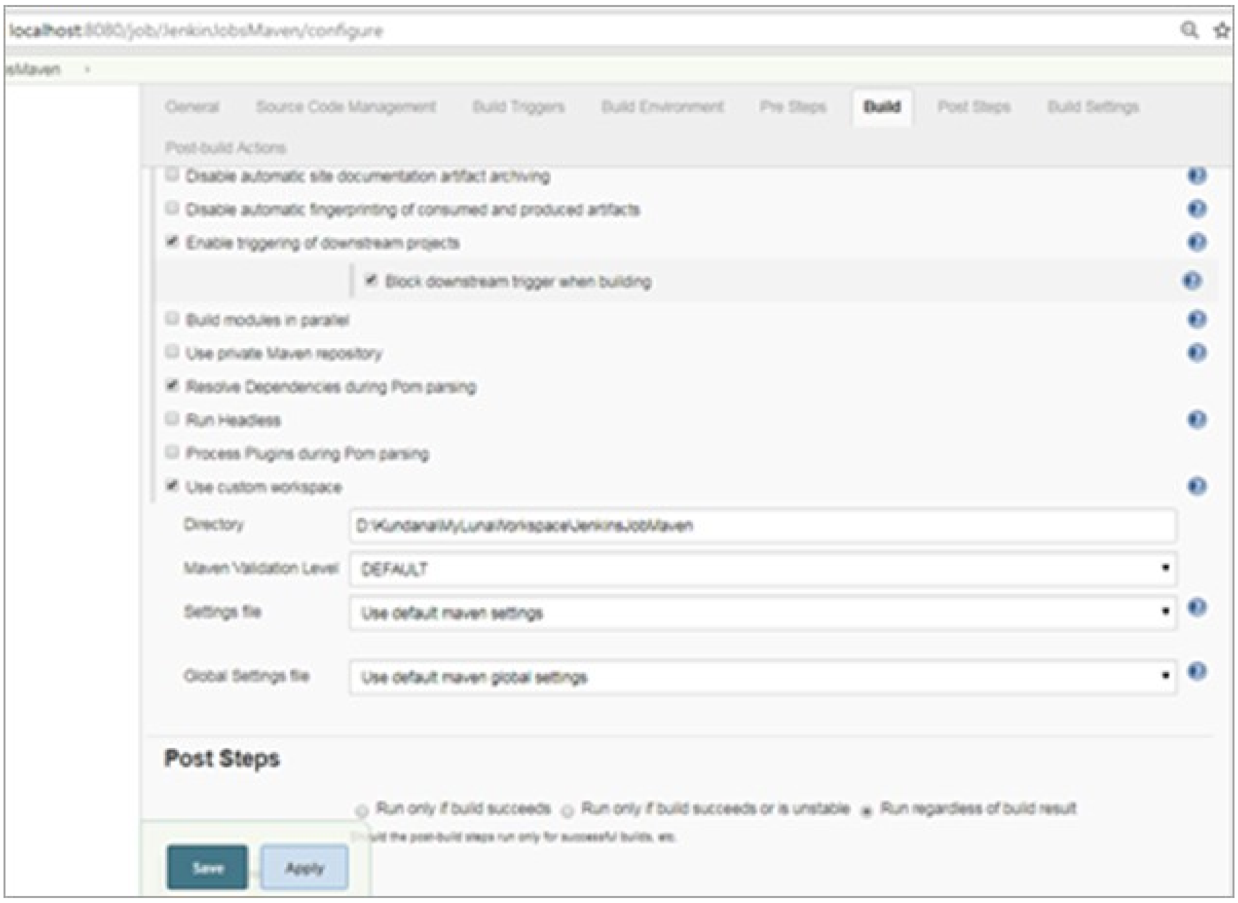
GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

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.



☐ Disable automatic site documentation artifact archiving
☐ Disable automatic fingerprinting of consumed and produced artifacts
☑ Enable triggering of downstream projects
☑ Block downstream trigger when building
☐ Build modules in parallel
☐ Use private Maven repository
☑ Resolve Dependencies during Pom parsing
☐ Run Headless
☐ Process Plugins during Pom parsing
☑ Use custom workspace

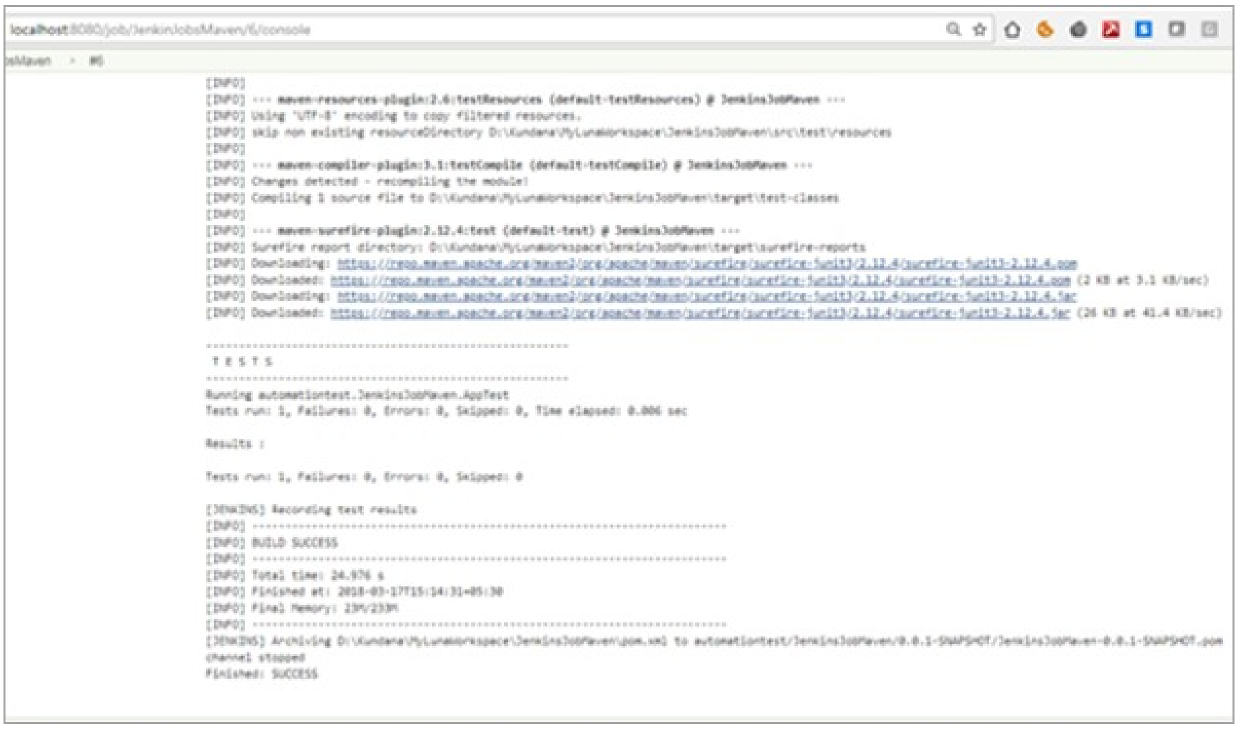
Directory	D:\Kundana\MyLuna\Workspace\JenkinsJobMaven
Maven Validation Level	DEFAULT
Settings file	Use default maven settings
Global Settings file	Use default maven global settings

Post Steps

○ Run only if build succeeds ○ Run only if build succeeds or is unstable ● Run regardless of build result

Should the post-build steps run only for successful builds, etc.

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.



```
[INFO]
[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:\Kundana\MyLunaWorkspace\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:\Kundana\MyLunaWorkspace\JenkinsJobMaven\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ JenkinsJobMaven ---
[INFO] Surefire report directory: D:\Kundana\MyLunaWorkspace\JenkinsJobMaven\target\surefire-reports
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.pom
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.pom (2 KB at 3.1 KB/sec)
[INFO] Downloading: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.jar
[INFO] Downloaded: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-junit3/2.12.4/surefire-junit3-2.12.4.jar (26 KB at 41.4 KB/sec)

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running automationtest.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

[JENKINS] Recording test results
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 24.976 s
[INFO] Finished at: 2018-03-17T15:14:31+05:30
[INFO] Final Memory: 23M/233M
[INFO] ------------------------------------------------------------------------
[JENKINS] Archiving D:\Kundana\MyLunaWorkspace\JenkinsJobMaven\pom.xml to automationtest/JenkinsJobMaven/0.0.1-SNAPSHOT/JenkinsJobMaven-0.0.1-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.