

Jenkins configuration with Maven and GitHub

Reference -<http://www.seleniumeasy.com/jenkins-tutorials/configure-maven-github-jenkins-to-run-testng.xml>

[Home](#) >> [Jenkins Continuous Integration Tutorial](#) >> [Jenkins configuration with Maven and GitHub](#)

Git is the most widely used modern version control system in the world today, which allows multiple persons to safely work on the same project without hampering other team members. As a part of a team using Git, You and your team members will clone working copy of a local repository from Git server. You/team will add and commit the test scripts that are developed locally and push your changes to the Git.

We can make Jenkins to pull the project's source code from the remote Git Server, by selecting the option in Source code management and specify the path/url can find the source code of the project.

In the below example, we will configure Git plugin and invoke testng.xml using maven from jenkins. You can have your selenium scripts/tests added to testng.xml. You can also parameterize your selenium tests [using Maven-Testng](#). View previous article on how to [configure simple maven project in jenkins](#)

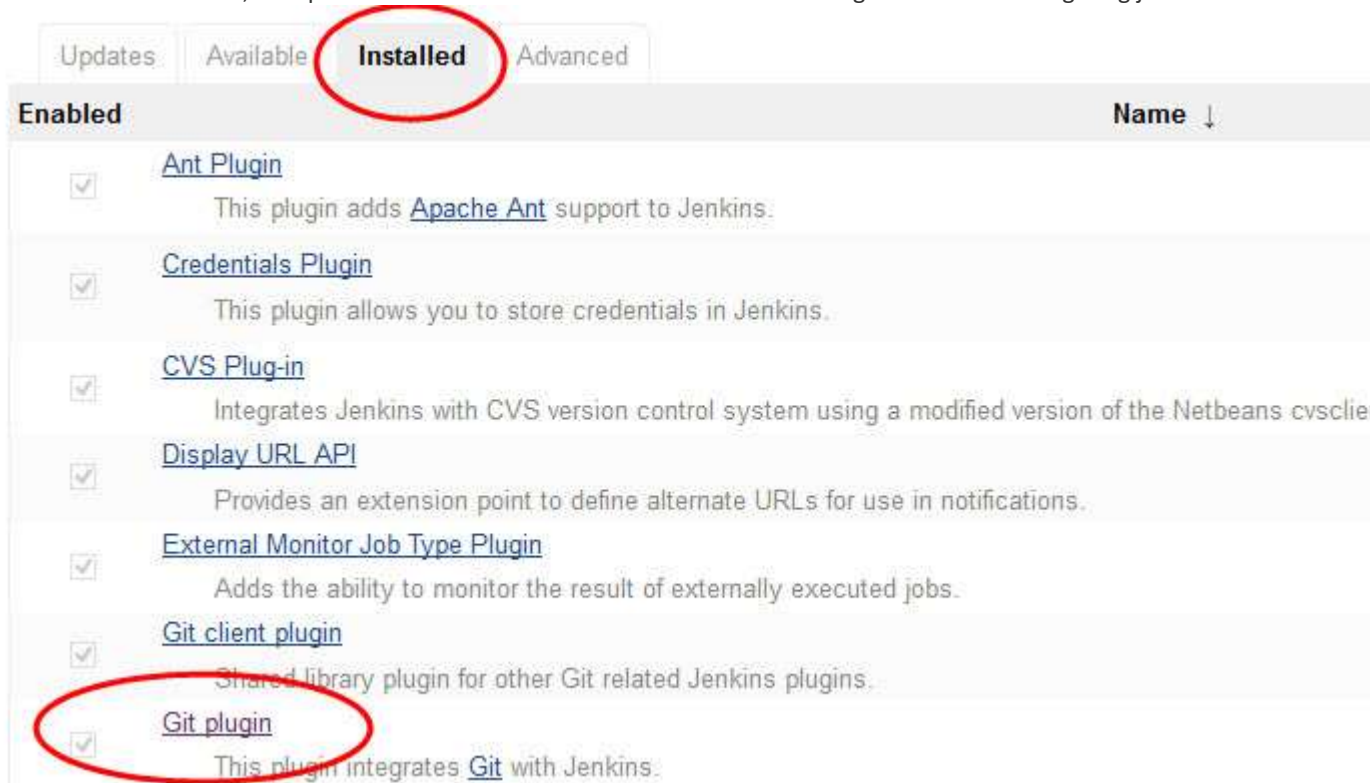
Configure Git Plugin in Jenkins

Step 1:- Manage Plugins -> Filter list of plug-ins available with 'Git Plugin'. Find more details about [Git Plug-in](#)

Step 2:- Check the Git Plug-in and click on the button 'Install without restart'

Step 3:- After the installations are done , Please restart Jenkins by using the command in the browser. <http://localhost:8080/jenkins/restart>

Once Jenkins is restarted, Git option should be available under source code management when configuring job.



Step 4:- From Manage Jenkins > Configure System, please provide the right Path to Git executable.

Git

Git installations

Name	
Path to Git executable	C:\Program Files\Git\bin\git.exe
<input type="checkbox"/> Install automatically	

Step 5:- Under Git Plug-in, Set the global git user.name and user.email to match your global config options

Git plugin



Global Config user.name Value: xxxxxxxx

Global Config user.email Value: xxxxxxxxxxxx@xxx.com

Create new accounts base on author/committer's email ☐

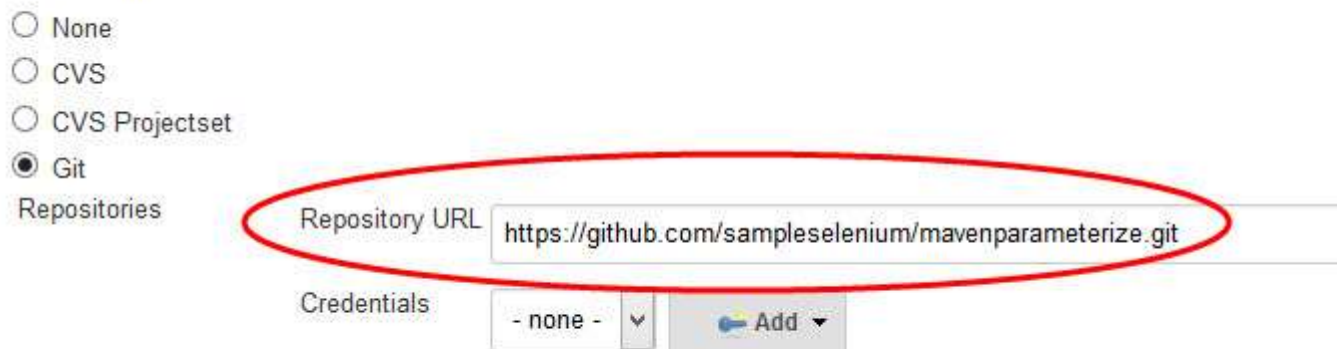
Now let us create a Maven Project and build a job from Git project

Step 1:- Click New Items -> Enter Project Name -> Select Maven Project -> Click OK

Step 2:- Provide the job description

Step 3:- In Source Code Management, Select 'Git' option. (This should be visible once you have successfully installed Git Plugin).

Source Code Management



☐ None

☐ CVS

☐ CVS Projectset

☒ Git

Repositories

Repository URL: https://github.com/sampleselenium/mavenparameterize.git

Credentials: - none - Add

Step 4:- From Build Triggers, If you want Jenkins to monitor the repository and start a build whenever any changes have been committed, We can choose to pick the Poll SCM option and enter syntax of cron.

The other options include Build periodically (for example, once a day,), Build whenever a SNAPSHOT dependency is built etc.

Here "SNAPSHOT" means build is still under active development. Ex;- If it is easy-1.0-SNAPSHOT.jar library, Maven will know that this version is not stable and is subject to changes. Before 1.0 release (or any other release) is done, there exists a 1.0-SNAPSHOT version.

The difference between a "real" easy-1.0 version and a easy-1.0-snapshot version is that snapshots might get updates. That means that downloading 1.0-SNAPSHOT might give you a different code than downloading it yesterday or tomorrow.

Builds Periodically will trigger builds as per the schedule (If we specify H/5 * * * *, every 5 minutes) even if you haven't changed anything. *Poll SCM* will check for changes before triggering any build, if there are changes to the previous version then only the build will be triggered.

Both these fields follows below syntax of cron which consists of 5 fields separated by TAB or white space: -

MINUTE (0-59), HOUR (0-23), DAY (1-31), MONTH (1-12), DAY OF THE WEEK (0-7)

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)

DAY OF THE WEEK - The day of the week (0-7) where 0 and 7 are Sunday.

Example:-

If you want to run job for every fifteen minutes, we should specify cron syntax as below :
H/5 * * * *

#If you want to trigger for, every 2 hours
H */2 * * * to distribute load evenly throughout the hour

In addition to the above, Jenkins also support convenient aliases as @yearly, @annually, @monthly, @weekly, @daily, @midnight, and @hourly.

Build Triggers

- ☐ Build whenever a SNAPSHOT dependency is built
- ☐ Trigger builds remotely (e.g., from scripts)
- ☐ Build after other projects are built
- ☒ Build periodically

Schedule

H/5 * * * *

To run job for every 5 minutes

[Optional]

Step 5:- In Build Settings, to send an email notifications, you can check 'Email Notification' and add Recipients address separated by comma.

Step 6:- In Post Build Actions, You can chose steps such as Archive Artifacts, Publish Results etc.

Step 7:- Click on Apply and Save.

Now as per configuration above, the build will trigger for every 5 minutes.

Once when the build is triggered, you can check for the console output, which will show you something like below image :-

```
Started by timer
Building on master in workspace C:\Users\VISISHTA\.jenkins\jobs\Sample Maven Project
> C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> C:\Program Files\Git\bin\git.exe config remote.origin.url https://github.com/sample/mavenparameterize.git
Fetching upstream changes from https://github.com/sample/mavenparameterize.git
> C:\Program Files\Git\bin\git.exe --version # timeout=10
> C:\Program Files\Git\bin\git.exe fetch --tags --progress https://github.com/sample
> C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/master^{commit}" #
> C:\Program Files\Git\bin\git.exe rev-parse "refs/remotes/origin/origin/master^{com
Checking out Revision c9723d6e38058bde3af717f365400e6924703999 (refs/remotes/origin/m
> C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
> C:\Program Files\Git\bin\git.exe checkout -f c9723d6e38058bde3af717f365400e6924703
> C:\Program Files\Git\bin\git.exe rev-list c9723d6e38058bde3af717f365400e6924703999
Parsing POMs
[workspace] $ "C:\Program Files\Java\jdk8\bin/java" -cp "C:\Users\VISISHTA\.jenkins\p
2.5.2.jar;C:\Program Files\maven\conf\logging" jenkins.maven3.agent.Maven31Main "C:\P
\.jenkins\plugins\maven-plugin\WEB-INF\lib\maven31-interceptor-1.5.jar C:\Users\VISIS
<===[JENKINS REMOTING CAPACITY]===>channel started
Executing Maven: -B -f C:\Users\VISISHTA\.jenkins\jobs\Sample Maven Project\workspac
```

And also when you view the build information, it will show you something like the below image which has information like who started and from where the build is triggered:-

Build #7 (Jan 29, 2017 5:11 PM)



No changes.



Started by timer



git

Revision: c9723d6e38058bde3af717f365400e6924703999

- refs/remotes/origin/master



[Test Result](#) (no failures)

Started automatically
by the timer

Module Builds

 [My Maven Project](#) 13 sec

As we are triggering build for every 5 mins, you can see the jobs running every 5 minutes (shows in the below image).

Build History		trend
find		
 #9	Jan 29, 2017 5:21 PM	
 #8	Jan 29, 2017 5:16 PM	
 #7	Jan 29, 2017 5:11 PM	

Hope this article helps you. Please let me know if you have any suggestions / issues.