

SE489 DevOps Engineering

Lab 5



Lab 5: Integration of Jenkins with Git

Overview: we will learn in this lab, how to configure Jenkins to perform Continuous Integration with Github. While Git take care of source code, Jenkins provides continuous integration. To demonstrate this, we will adopt following strategy-

- a. We will first create a github repository
- b. We will then push some files into it
- c. We will then create a Jenkin job
- d. We will make changes into the source code
- e. Push this changed source code file to github repository
- f. Will observe the subsequent effects by Jenkins

1. Let's create a sample java source code file, and put it into the local working directory

```
JenkinsWithGit.java - Notepad
File Edit View
package miscellaneous;

/**
 *
 * @author Zafar Iqbal Khan, zkhan@psu.edu.sa
 */
public class JenkinsWithGit {
    public static void main(String[] args) {
        //code to print Unicode characters
        int charPerLine=0;
        for (int i = 33; i < 256; i++, charPerLine++)
        {
            System.out.print((char)i+((charPerLine%10==0)?"\n":"\t"));
            System.out.print();
        }
    }
}
```

Ln 1, Col 1 | 70% | Unix (LF) | UTF-8

- From the master directory, check status

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    DevOps/
    JenkinsWithGit.java

nothing added to commit but untracked files present (use "git add" to track)
mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

- Now, let's push JenkinsWithGit.java to staging area, with add command, and then check status again

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git add JenkinsWithGit.java

mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   JenkinsWithGit.java

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    DevOps/

mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

- Now commit this file for push operation

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git commit -a -m "Pushing JenkinsWithGit.java"
[master bbe19b8] Pushing JenkinsWithGit.java
1 file changed, 18 insertions(+)
create mode 100644 JenkinsWithGit.java

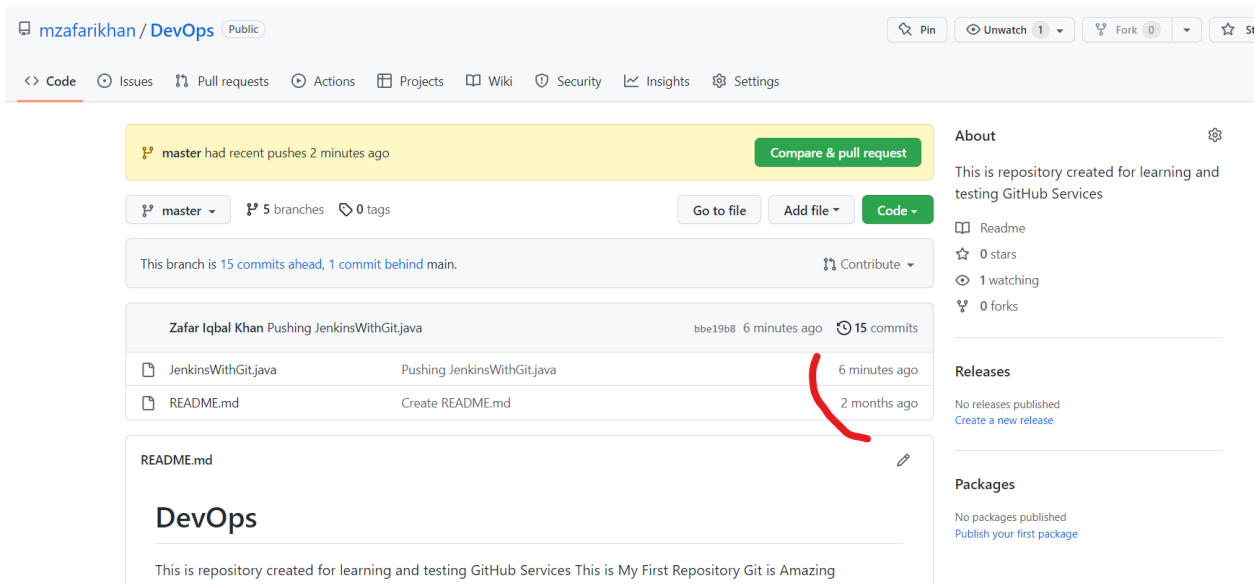
mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

5. Let's push JenkinsWithGit.java to the global repository

```
MINGW64~/d/DevOps Tools/Lab Manual
mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 569 bytes | 284.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mzafarikh/DevOps.git
 558211c..bbe19b8  master -> master

mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

6. Cross check this on GitHub website



7. Before we create a Jenkins job, first check if the Git plugin is installed in the Jenkins, if it is not, we have to install it first, Open any web browser and enter url <http://localhost:8080> , Jenkins

dashboard will open, go to manage Jenkins

The screenshot shows the Jenkins dashboard interface. The left sidebar contains navigation options: New Item, People, Build History, Manage Jenkins (highlighted), My Views, and New View. Below the sidebar are sections for Build Queue (empty) and Build Executor Status (2 idle). The main content area features a table with columns: S, W, Name, Last Success, Last Failure, and Last Duration. A single row is visible for 'FirstJenkinsProject'.

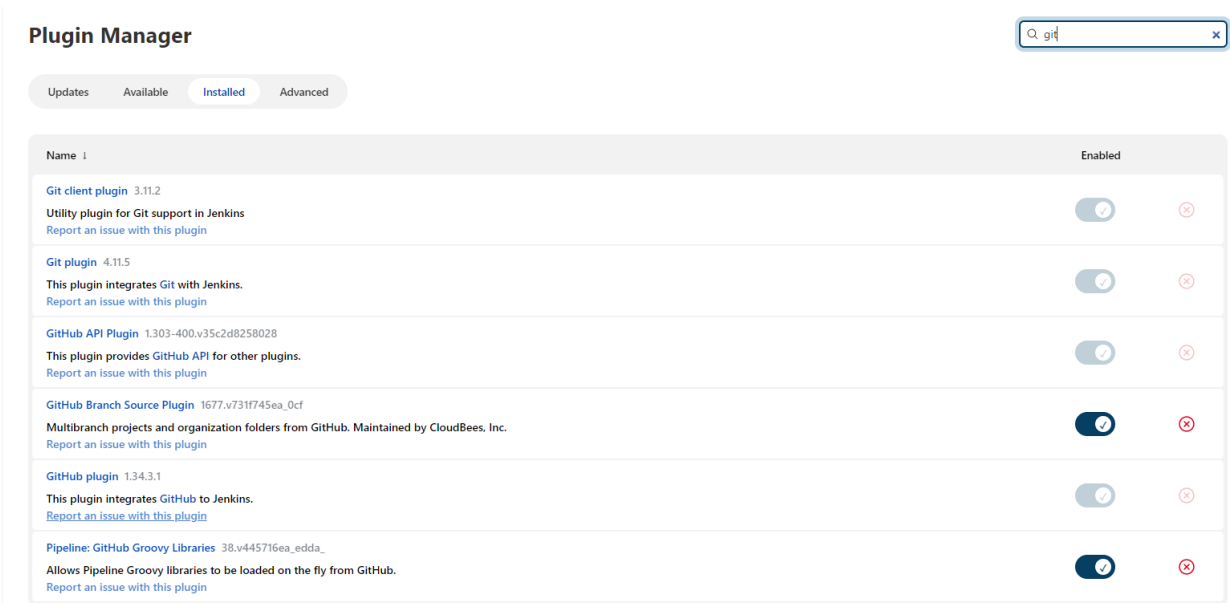
S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	⚙️	FirstJenkinsProject	15 days #3	N/A	1.1 sec

Then click on, **manage plugins**

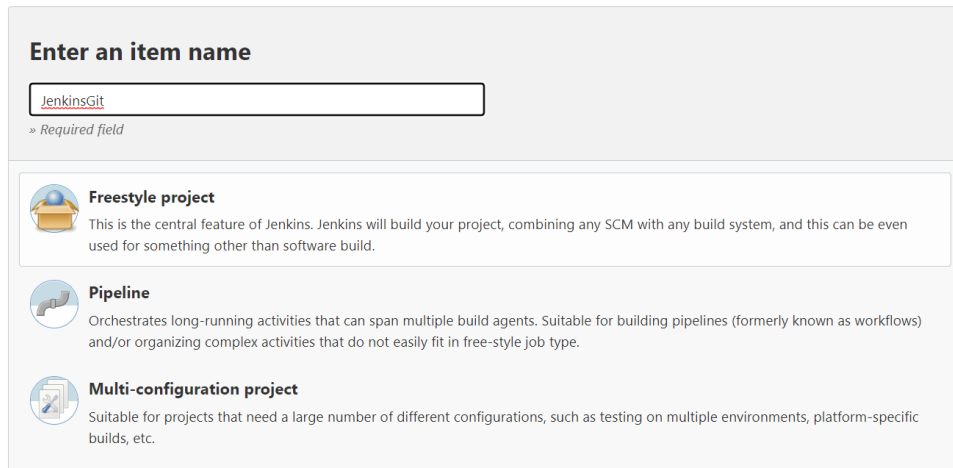
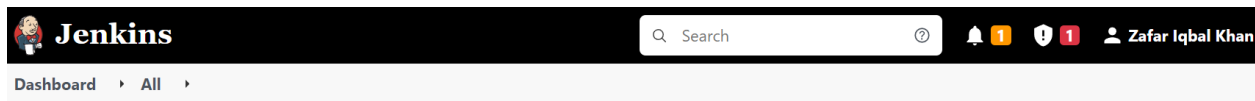
The screenshot shows the 'Manage Jenkins' configuration page. It includes a notification for a new Jenkins version (2.361.1) and a warning about security vulnerabilities. Under the 'System Configuration' section, the 'Manage Plugins' option is circled in red. Other options include Configure System, Global Tool Configuration, and Manage Nodes and Clouds. The 'Security' section includes Configure Global Security, Manage Credentials, Configure Credential Providers, and Manage Users.

On subsequent screen, look for **Git plugin**, if it is there, OK, Otherwise, search for it, and install it.

As in our case, it is already installed



8. Now let's create a new **Jenkins Item**, give it a name, and select type as **Freestyle Project** and click Ok



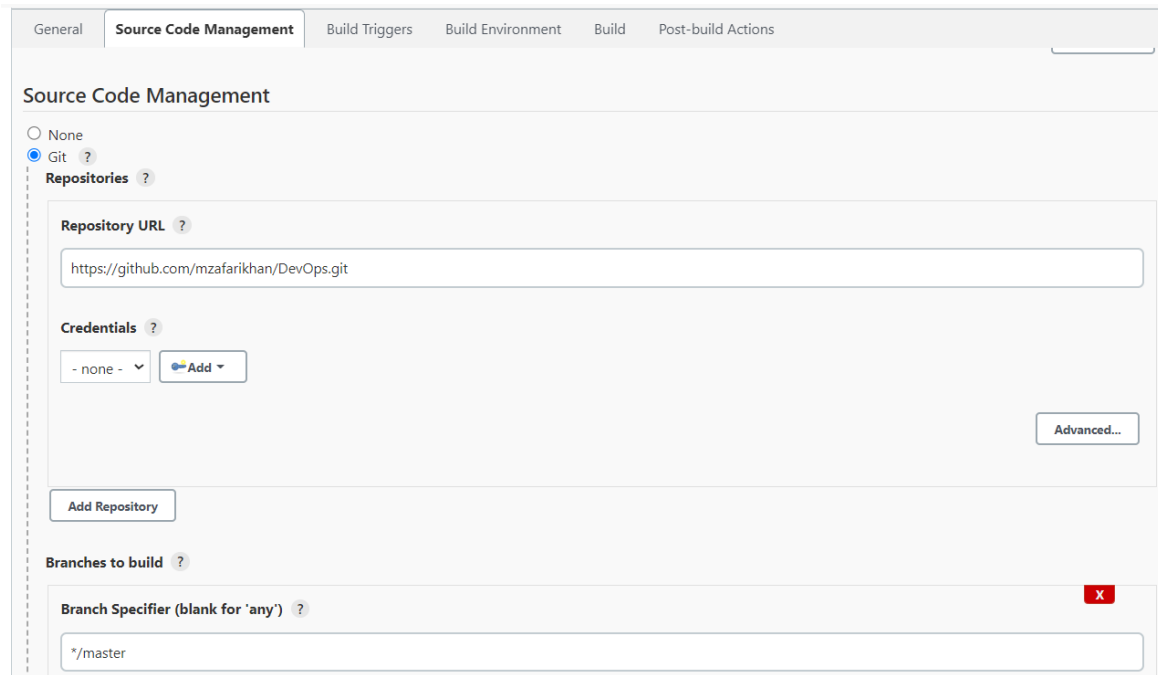
9. On subsequent screen, click on Source Code Management tab, choose Git, and provide URL** for the global repository, select Poll SCM under Build Triggers, provide schedule, ****/2 * * * ****

//we want Jenkins to Poll after every 2 minutes, 2 is here an offset value

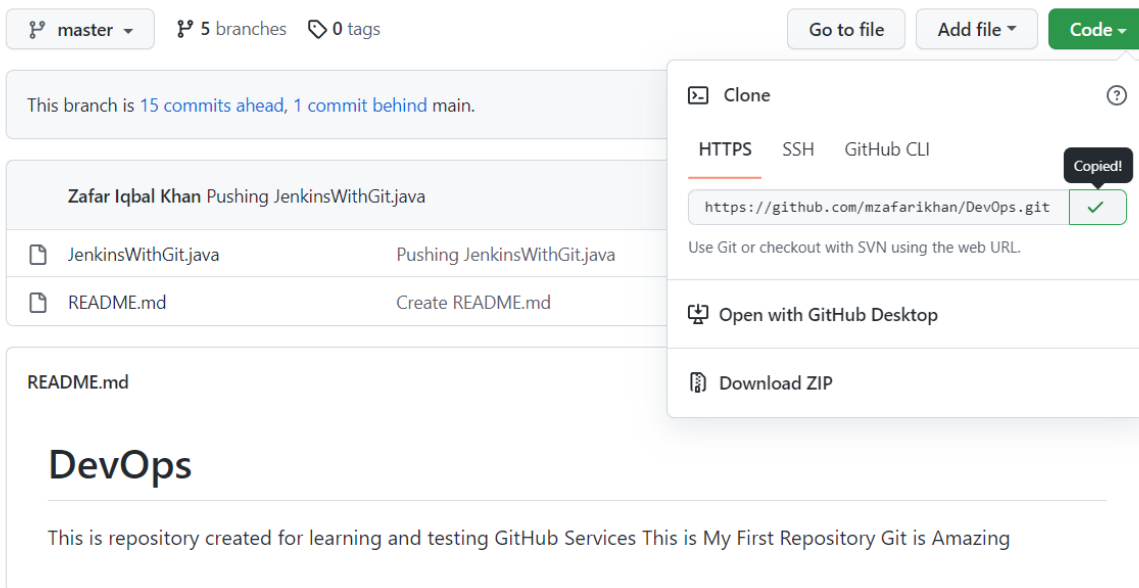
Hint: Scheduling in Jenkins follows standard CRON syntaxes

sample CRON schedule syntax is

{Minute} {Hour} {DayOfMonth} {Month} {DayofWeek}



**to know the URL of the global repository, login into GitHub account, browse to the required repository, click on Code tab, url to the repository is available there.



At the end, click on apply, and all is set now.

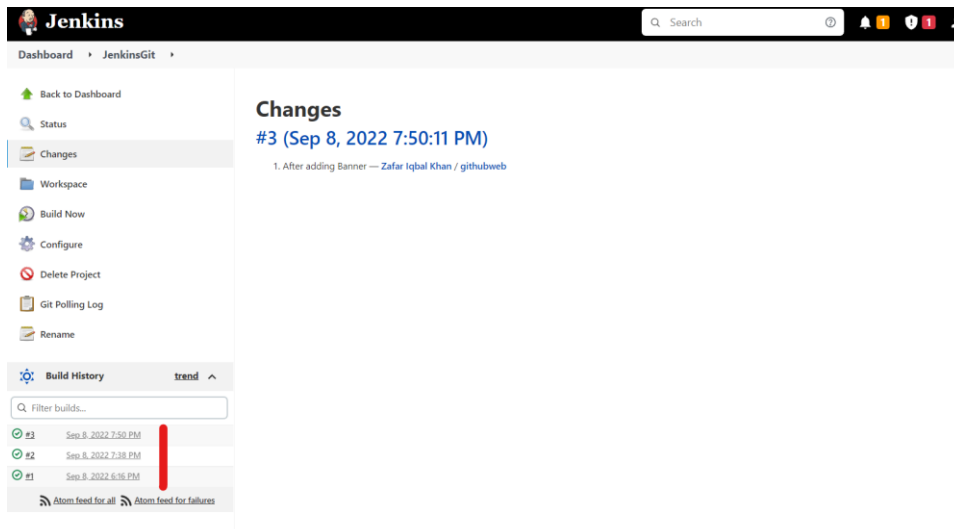
10. Now, make some changes into the *JenkinsWithGit.java* file and repeat steps from 2 – 5, i.e. after changes, push this file into Global repository.

```
MINGW64:/d/DevOps Tools/Lab Manual
mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git commit -m "After adding Banner"
[master 6f67296] After adding Banner
 1 file changed, 2 insertions(+), 1 deletion(-)

mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$ git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 381 bytes | 381.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:mzafarikhan/DevOps.git
 bbe19b8..6f67296  master -> master

mzafa@BlackHole MINGW64 /d/DevOps Tools/Lab Manual (master)
$
```

11. Now let's go to the Jenkins Dashboard, wait for at least, 2 minutes (as we have specified Polling schedule for every 2 minute), click on changes at the left pane, within 2 minutes, we will see, Jenkins starts a new build, and changes are listed automatically.



This is how continuous integration is achieved through GitHub and Jenkins