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

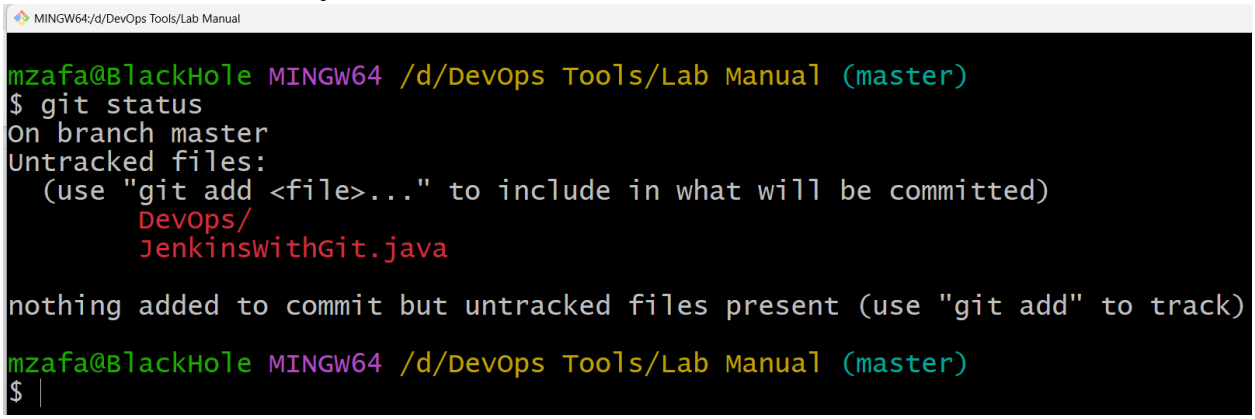
JenkinsWithGit.java - Notepad

File Edit View

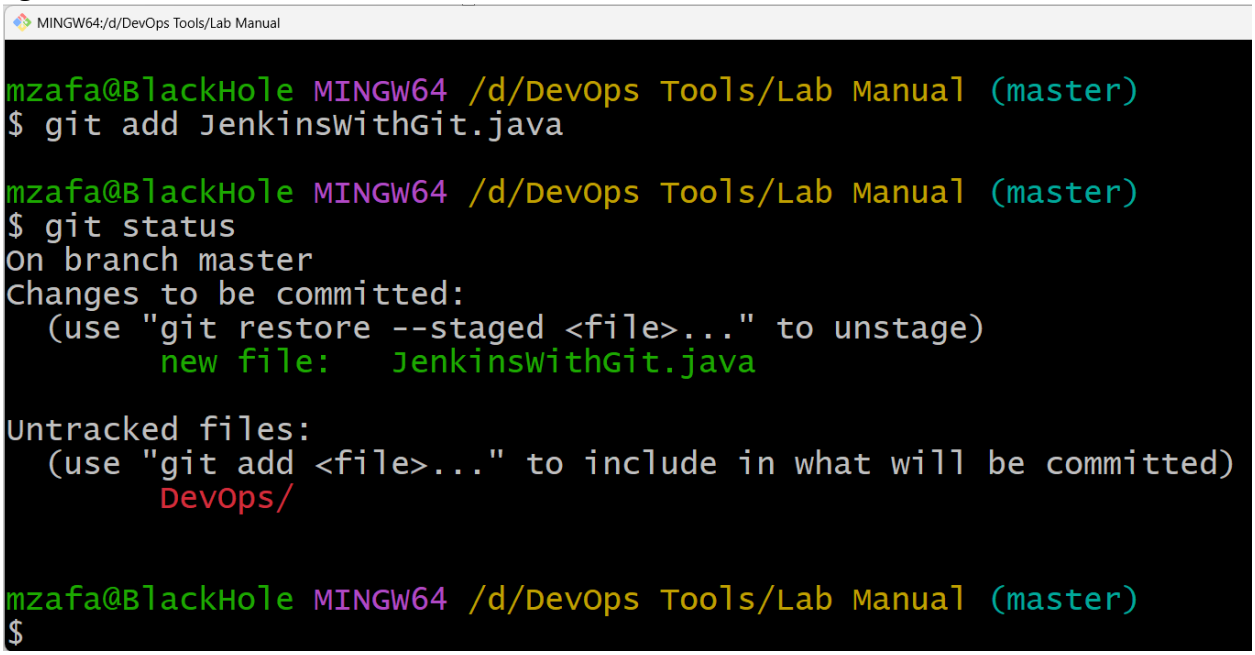
```
package miscelleneous;

/**
 *
 * @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

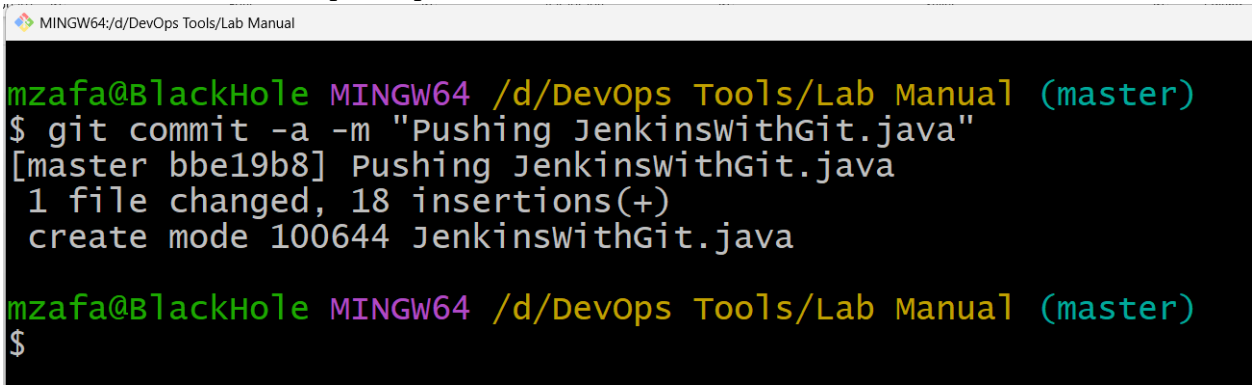
2. From the master directory, check status



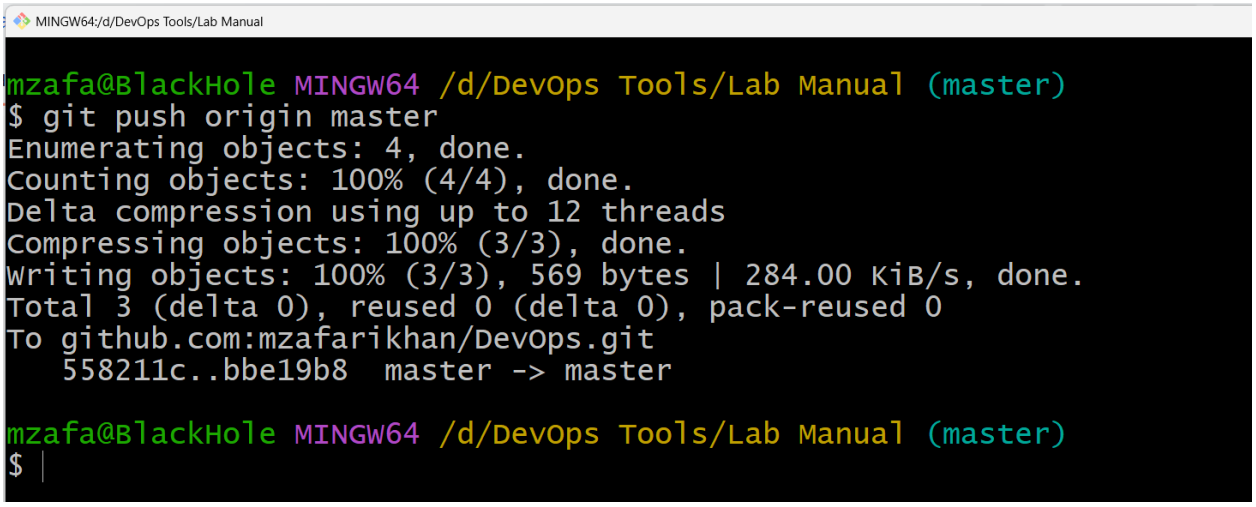
3. Now, let's push JenkinsWithGit.java to staging area, with add command, and then check status again



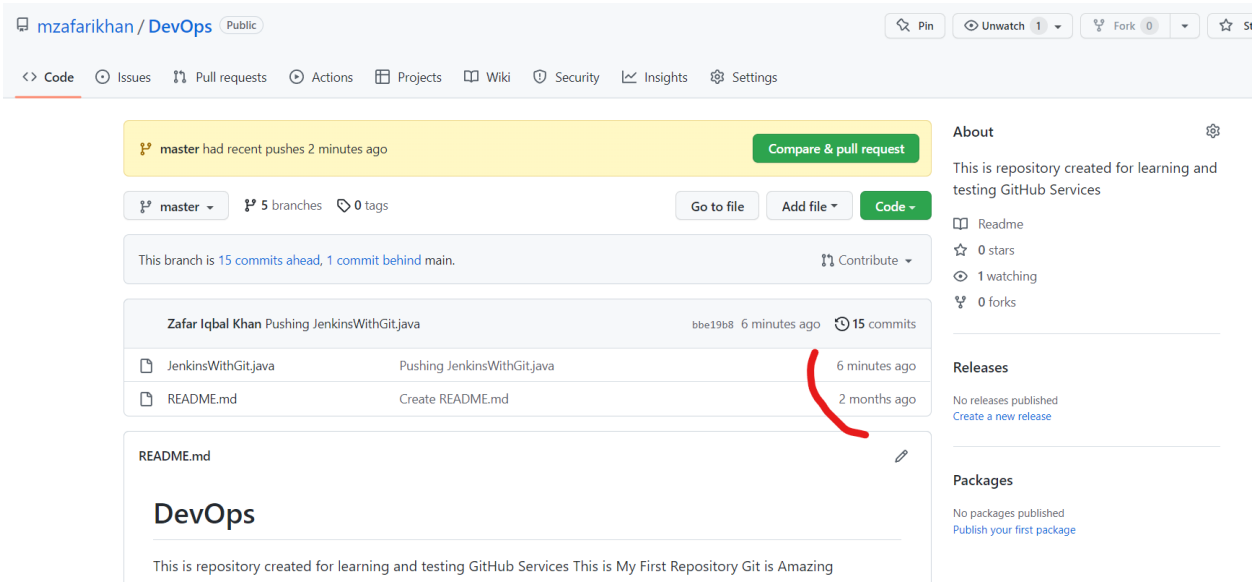
4. Now commit this file for push operation



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

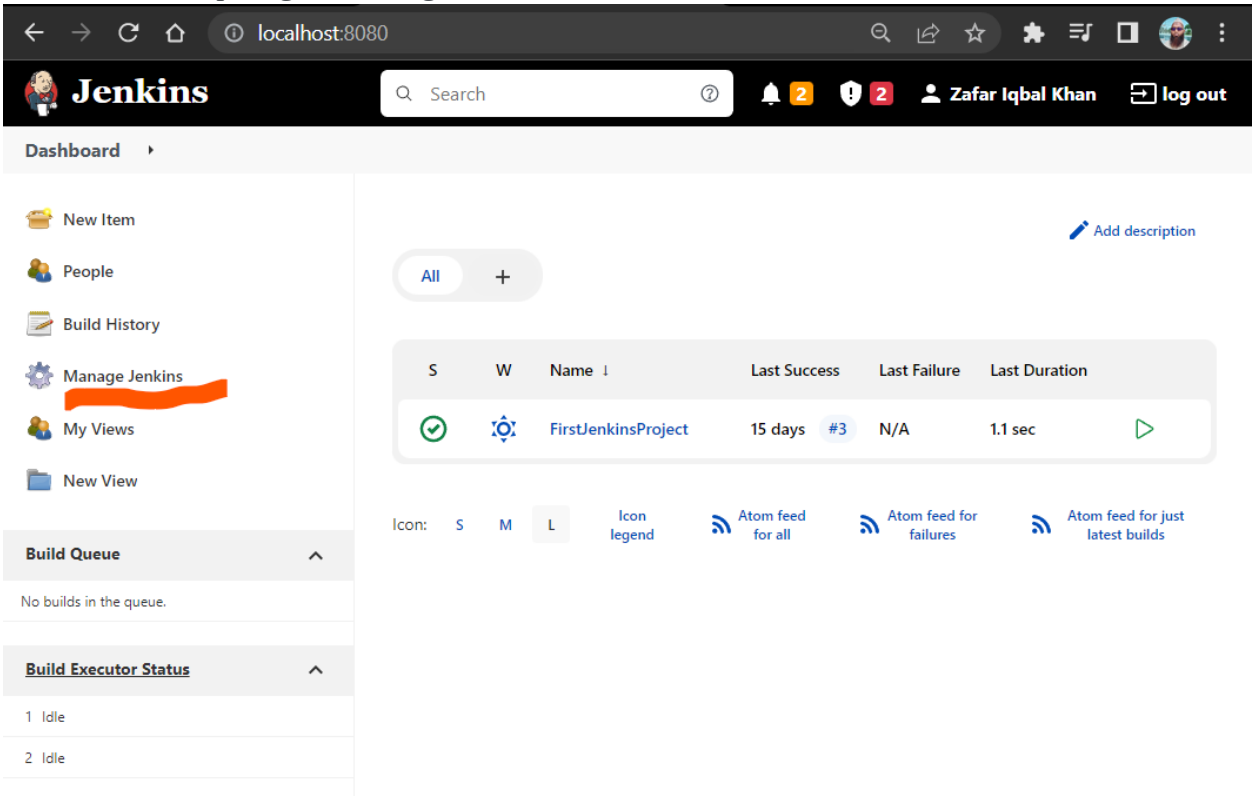


6. Cross check this on GitHub website

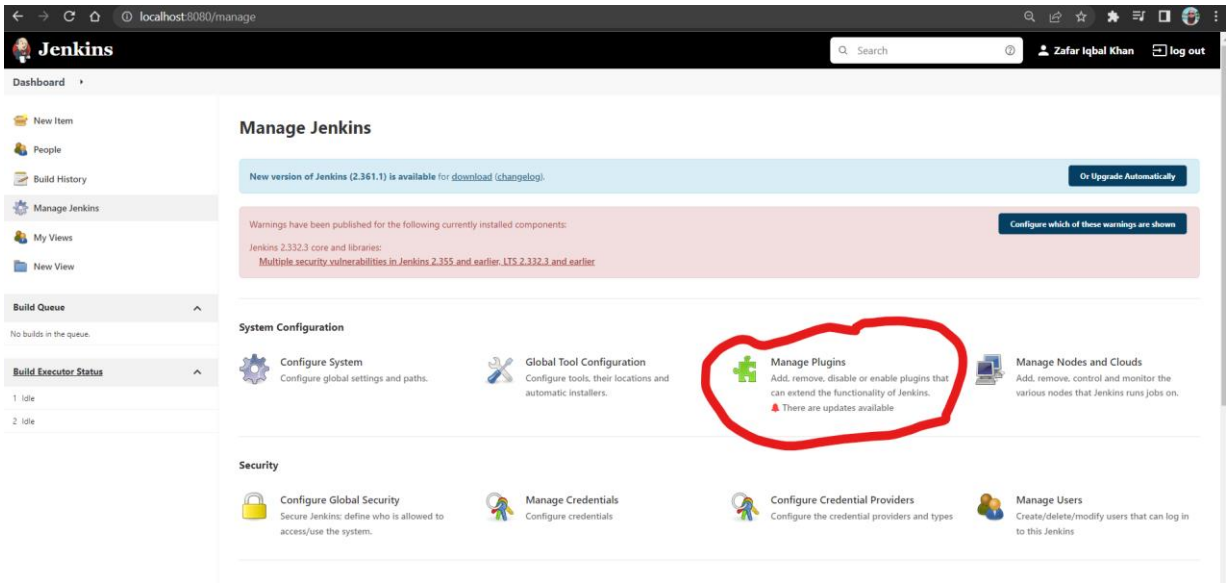


7. Before we create a Jenkins job, first check if the Git plugin is installed in the Jenkins, if it is not, we have to install it first, Open any web browser and enter url http://localhost:8080 , Jenkins

dashboard will open, go to manage Jenkins

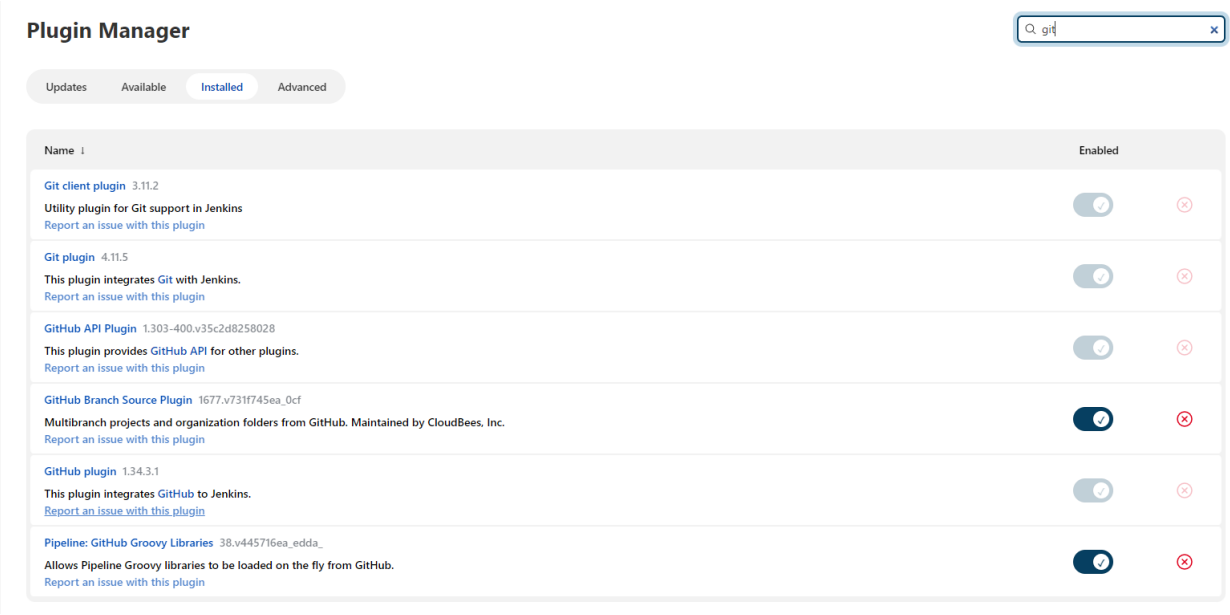


Then click on, **manage plugins**

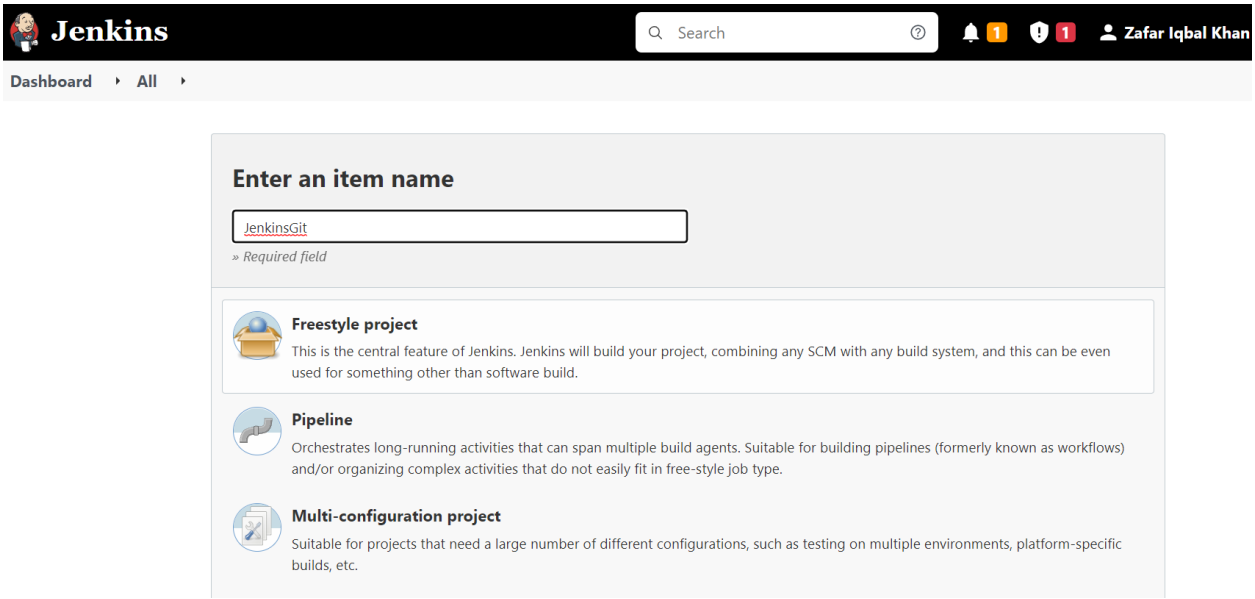


On subsequent screen, look for **Git plugin**, if it is there, OK, Otherwise, search for it, and install it.

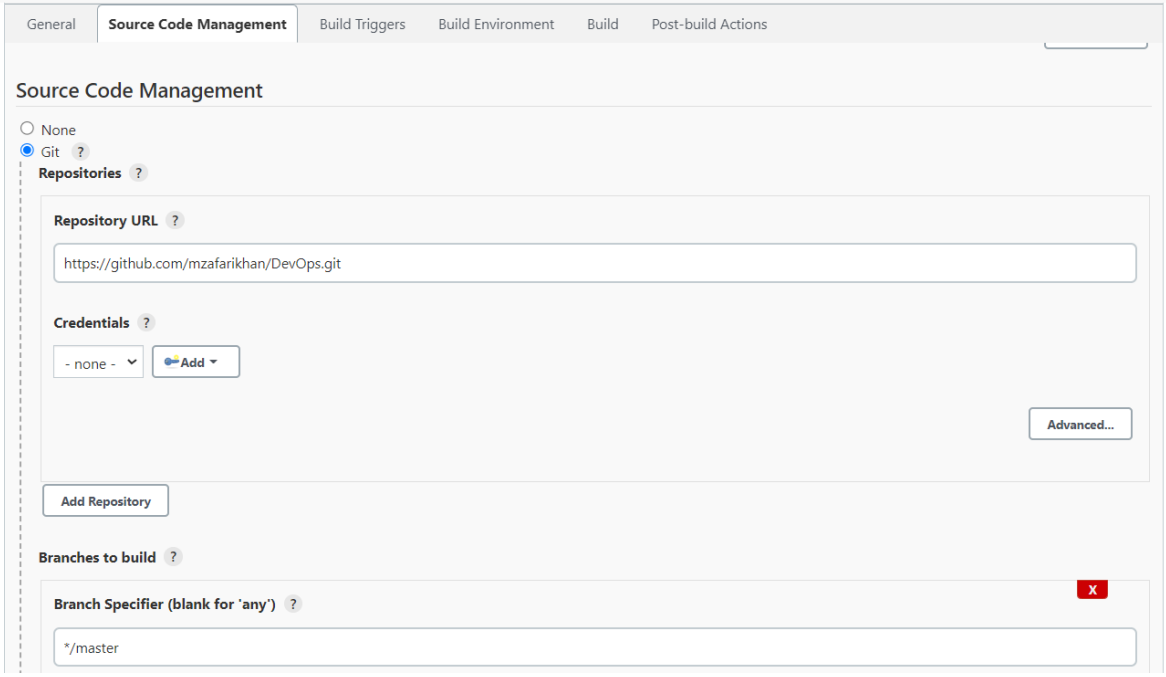
As in our case, it is already installed



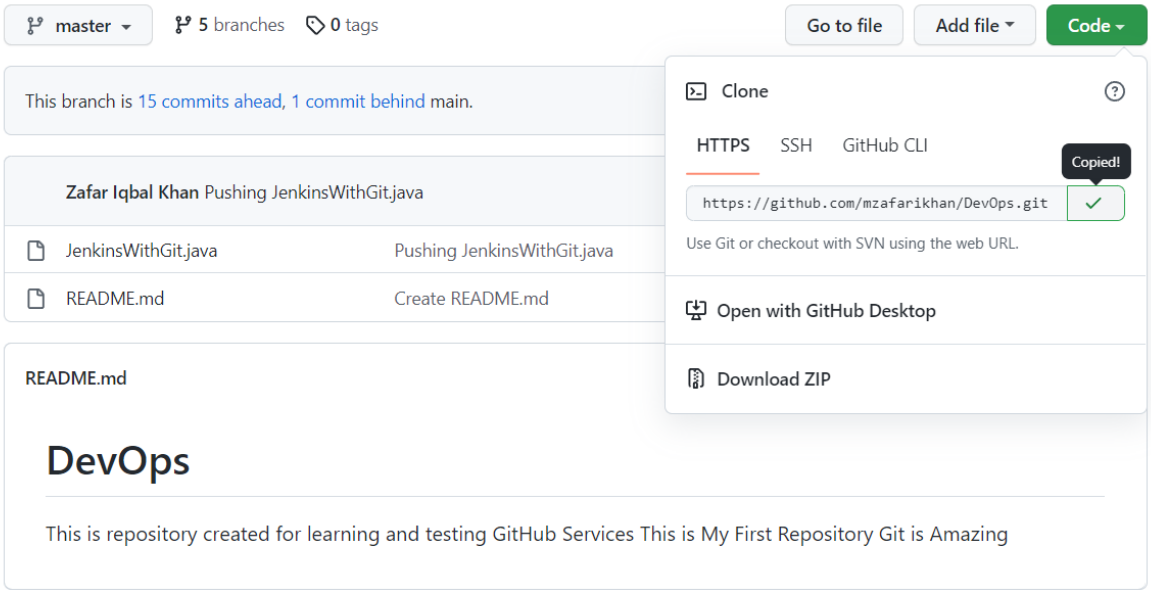
8. Now let's create a new **Jenkins Item,** give it a name, and select type as **Freestyle Project** and click Ok



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}

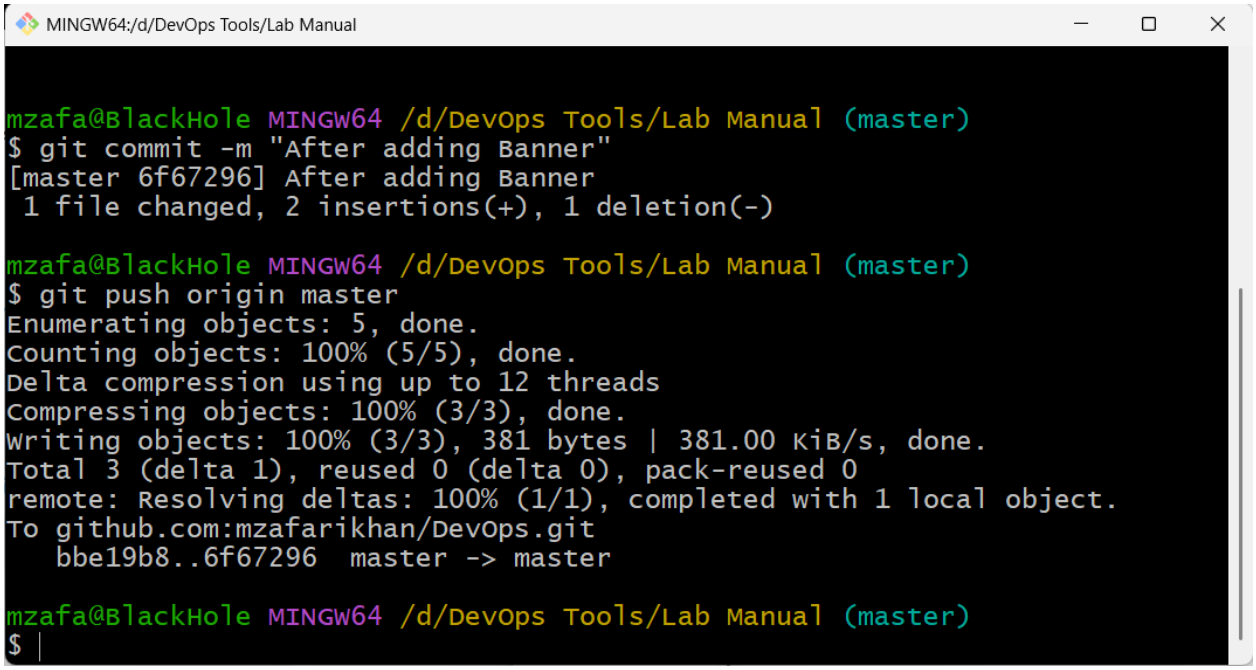


**to know the URL of the global repository, login into GitHub account, browse to the required repository, click on Code tab, url to the repository is available there.

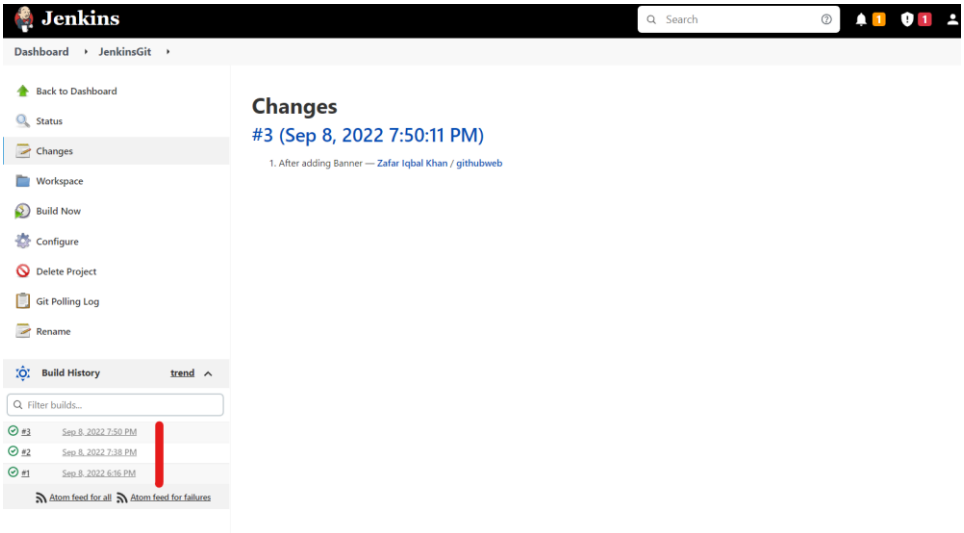


At the end, click on apply, and all is set now.

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.



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