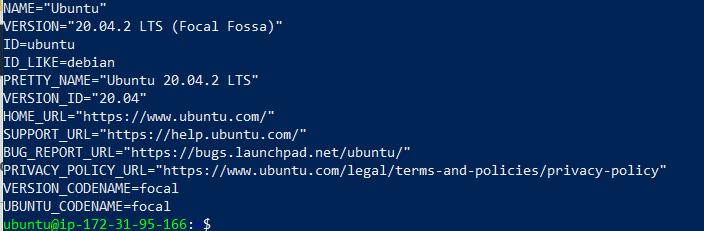
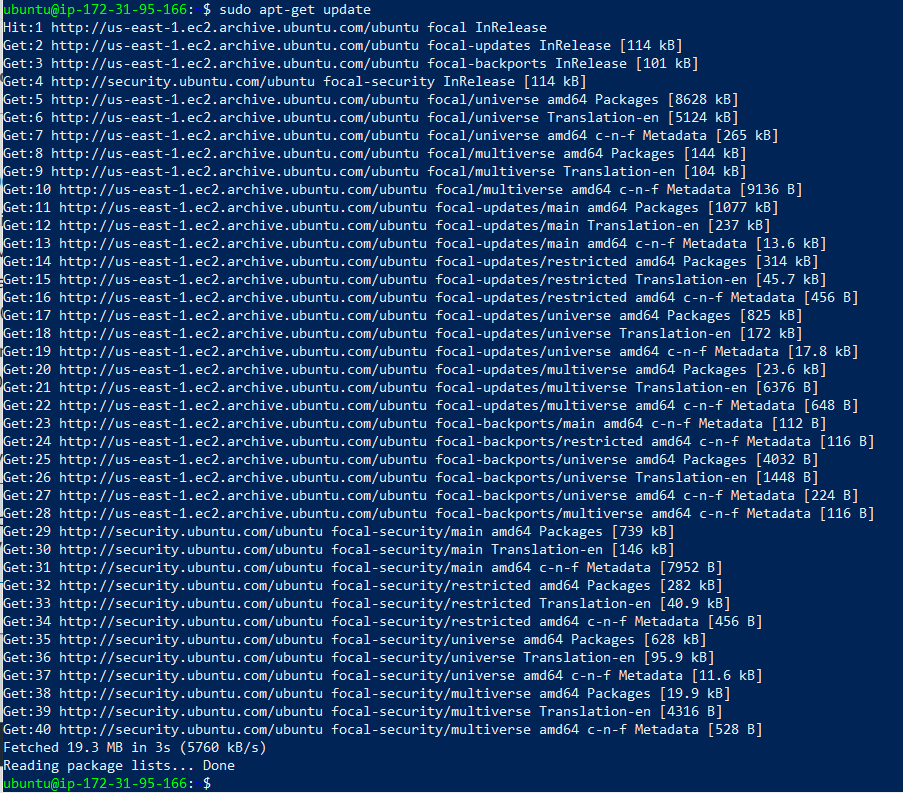
* From Terminal/Putty/Open the Linux machine check the which OS release.

**$ cat /etc/os-release**



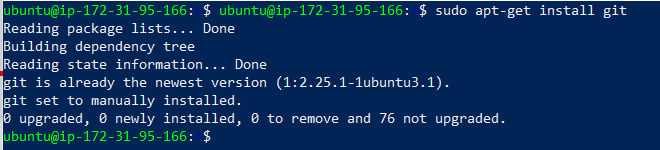
* Best practice that whenever you open the instance first time type the below command If the instance is already running no need to type this command.

**$ sudo apt-get update**



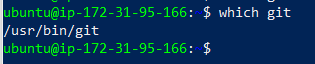
* Goto this site https://git-scm.com/download/linux and copy the git install command.

**$sudo apt-get install git**



* To check the path where the GIT is installed.

**$which git**



* To check GIT version.

**$git –version**

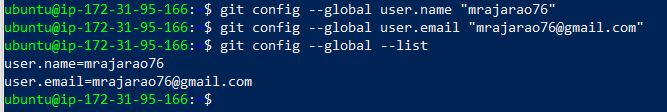


* This is one time activity for checkpoints team member names must be provided.(This is mandatory)

**$git config --global user.name "mrajarao76"**

**$ git config --global user.email "mrajarao76@gmail.com"**

**$ git config --global --list**



* Create the folder

**$mkdir chat-bot**

* Put the file in the folder

**$cd chat-bot**

Open Vi editor and copy the text file.

**Java "Hello, World!" Program**

// Your First Program

class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

// Your First Program

class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

To save the file Esc:wq! Or :wq zz

Without saving the file Esc:q!

Yank (copy a line of text) YY

Central Repository: When the repo is located on the VCS server

Local Repository: When the repo is located on the dev laptop

**GIT Life cycle:**

1. Working directory
2. Staging Area
3. Commit

**Working directory**

* The place where your project resides in your local disk.
* This project may or may not be tracked by **git**
* In either case the directory is called the working directory
* The project can be tracked by git, by using the command **git init**
* By doing **git init**, it Automatically creates a hidden. git folder

**Staging Area**

* Once we are in the working directory, we have to specify which files are to be tracked by **git**
* We do not specify all files to be tracked in **git**, because some files could be temporary data which is being generated while execution.
* To add files in the staging area, we use the command **git add.**

**Commit**

* Once the files are selected and are ready in the staging area, they can now be saved in repository.
* Saving a file in the repository of git is known as doing a commit.
* When we commit a repository in git, the commit is identified by commit id.
* The command for initializing this process is **git commit -m “message”.**

**Working directory**

* Initializing the git this is the working directory where the folder is created and files are stored in the local laptop

**$git init**

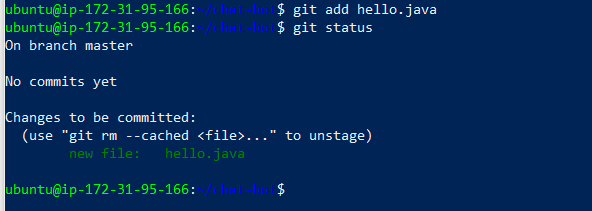


**Staging Area**

* Mark the files in the staging Areas with the below command

**$git add hello.java**

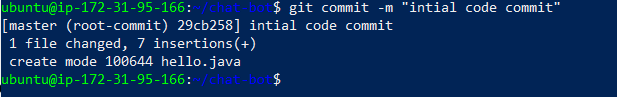
**$git status**



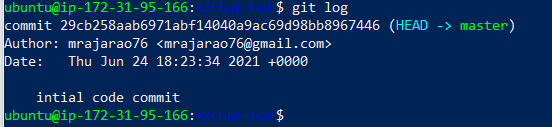
**commit**

* Commit the files Mark the files in the staging Areas with the below command

**$git commit -m “Initial code commit”**



**$git log**



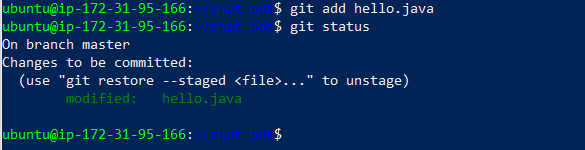
Git status will compare between working directory === .git (version history)

Hello.java == hello.java

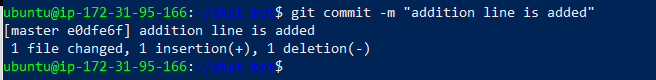
Modify the lines in the hello.java file and repeat the git life cycle

**$git add hello.java**

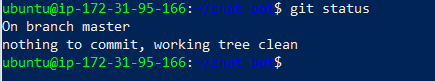
**$git status**



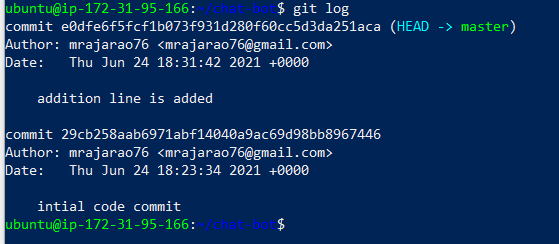
**$git commit -m “Added comments the lines in hello.java”**



**$git status**



**$git log**



GIT

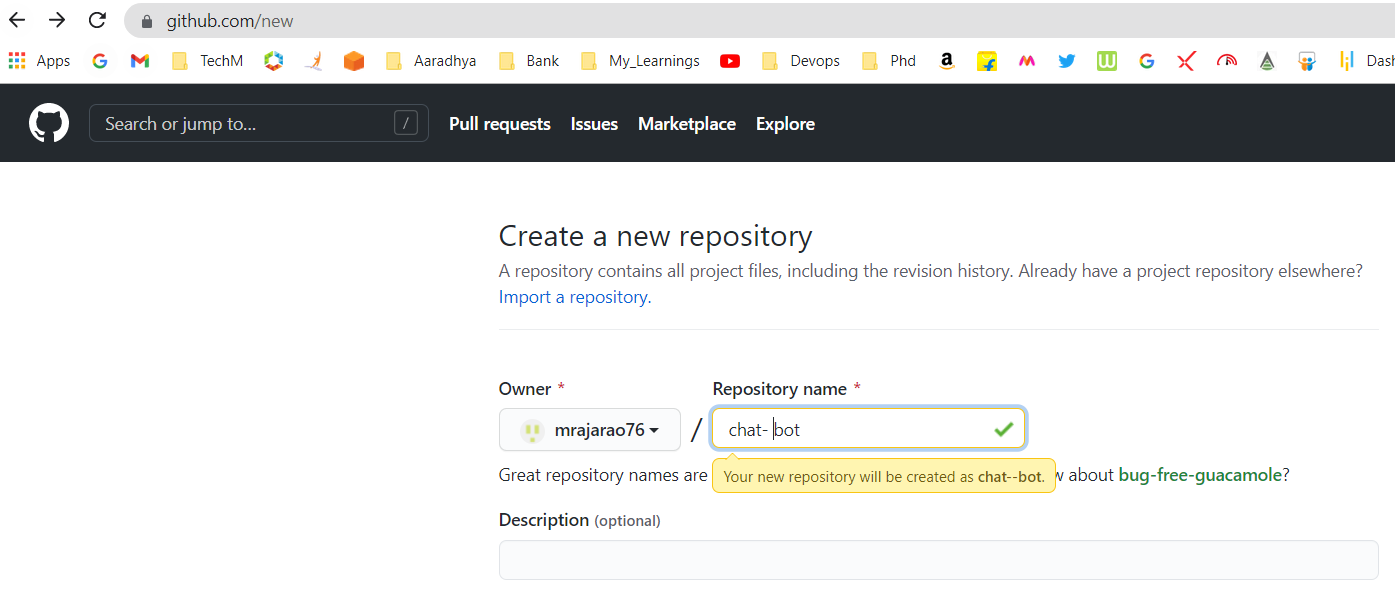
Features GIT (open source ) is lacking

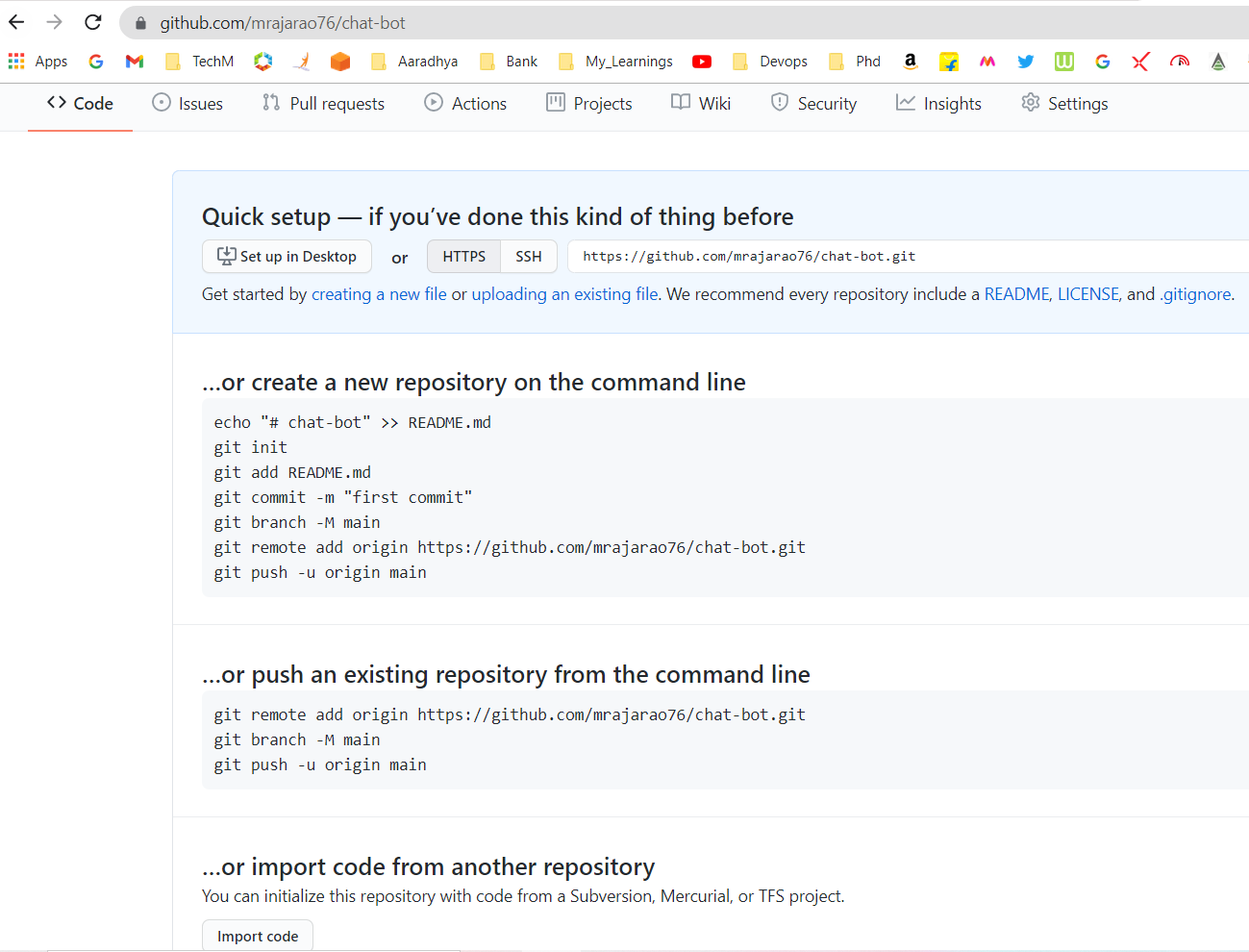
1. Lacks UI
2. LDAP integration - Authentication
3. RBAC- Role Based Access Control Authorization
4. Auditing – Who logged in when
5. High Availability -Clustering
6. Notifications
7. Third party tool integrations
8. Backup & Restore

To overcome this they have come up with centralized repository called

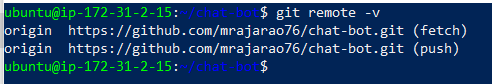
1. GitHub: https://github.com/
2. GitLab: https://about.gitlab.com/
3. Bitbucket: https://bitbucket.org/product

These are called the enterprise GIT



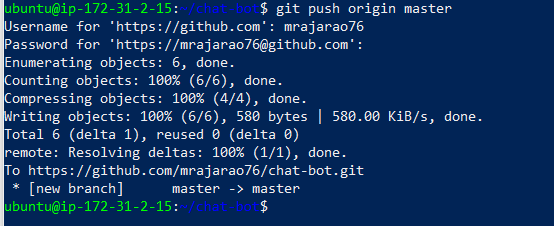


**$git remote -v**

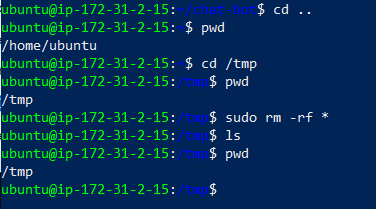


**$git remote add origin** [**https://github.com/mrajarao76/chat-bot.git**](https://github.com/mrajarao76/chat-bot.git)

**$git push origin master**

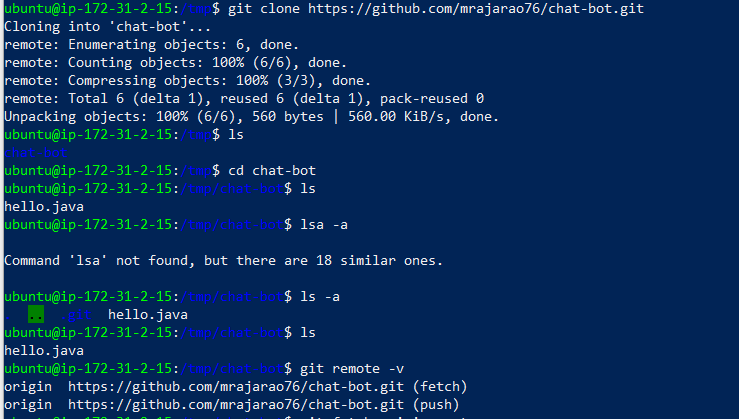


**Change the directory to tmp and clone chat-bot to tmp**

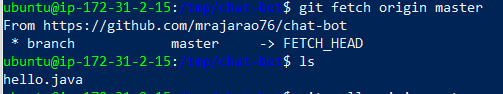


**Clone**

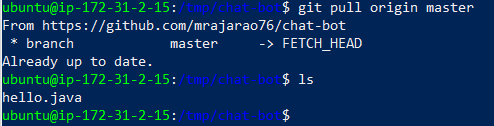
**$ git clone** [**https://github.com/mrajarao76/chat-bot.git**](https://github.com/mrajarao76/chat-bot.git)

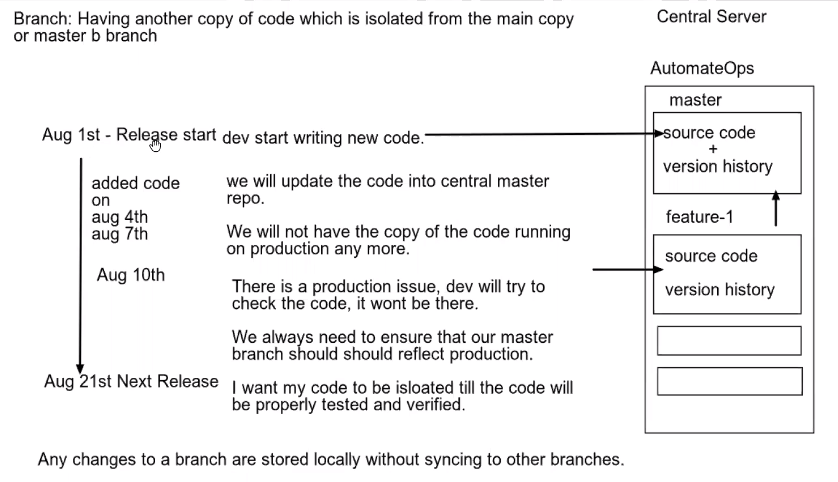


**$ git fetch origin master**



**$ git pull origin master**



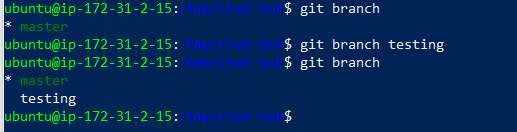


**$ git branch**



Create a new branch testing

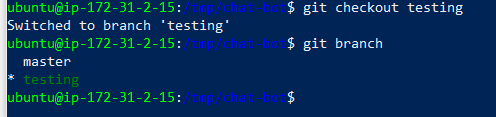
**$ git branch testing**



Switch to branch testing

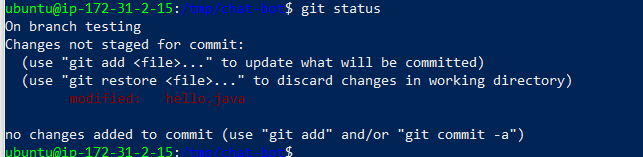
**$ git checkout testing**

**$ git branch**



Now we are in testing branch and make changes to the hello.java file an new line is added as “new branch testing” on this is edited in vi

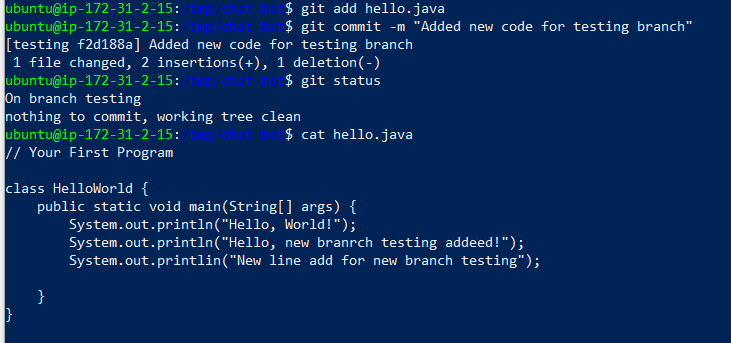
**$ git status**



Now git life cycle stage the area and commit

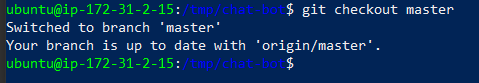
**$ git add hello.java**

**$ git commit -m “Added new code for testing branch”**



Now switch to master branch

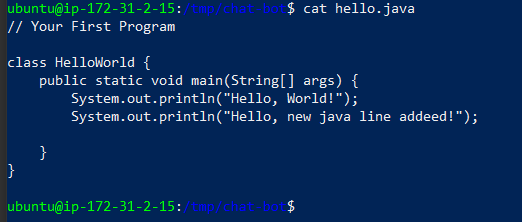
**$ git checkout master**



Now you are in master branch and view the hello.java

There is a difference in testing and master branch hello.java files

**$ cat hello.java**



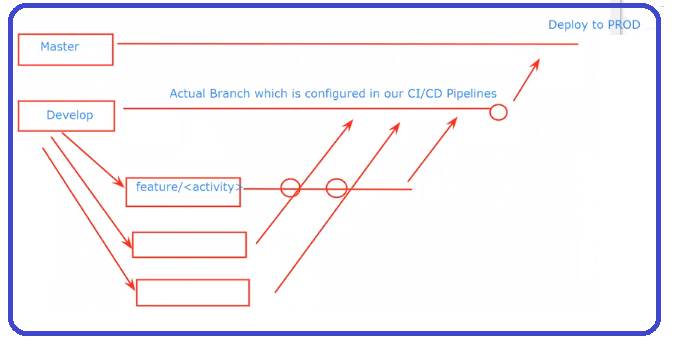
**Branching strategy:**

Rules on when to create a branch

When to merge and couple of other collaborative decisions

Git is very fexible and powerful

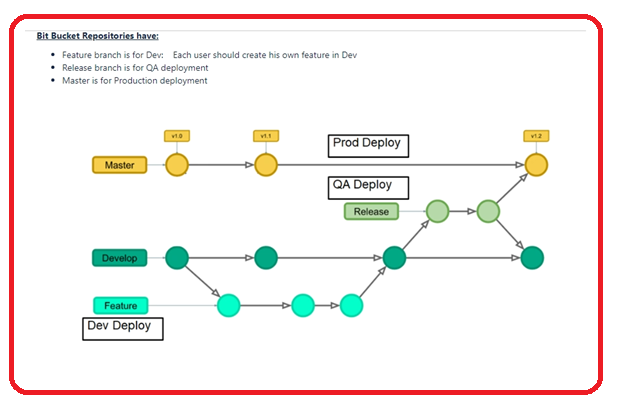
1. **Feature Branching Strategy (only)**
2. **GIT flow Branching Strategy**

****

**develop = non prod, Dev, SIT and UAT**

**Master = prod**

**Real Time FPL Bitbucket Repositories structure**



Now remove everything from chat-bot.

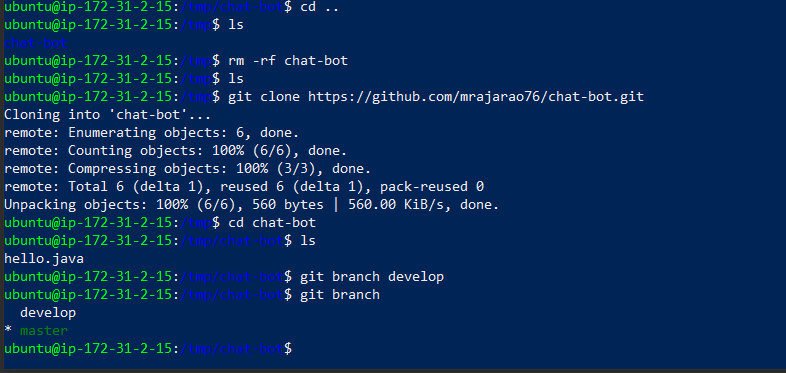
**$ rm -rf chat-bot**

Clone the chat-bot and create a new branch develop

**$ git clone https://github.com/mrajarao76/chat-bot.git**

**$ git branch develop**

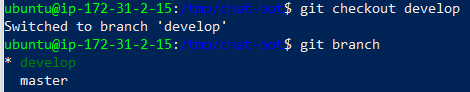
**$ git branch**



After cloning now switch to develop branch

**$ git checkout develop**

**$ git branch**



Now push the code to the central repository

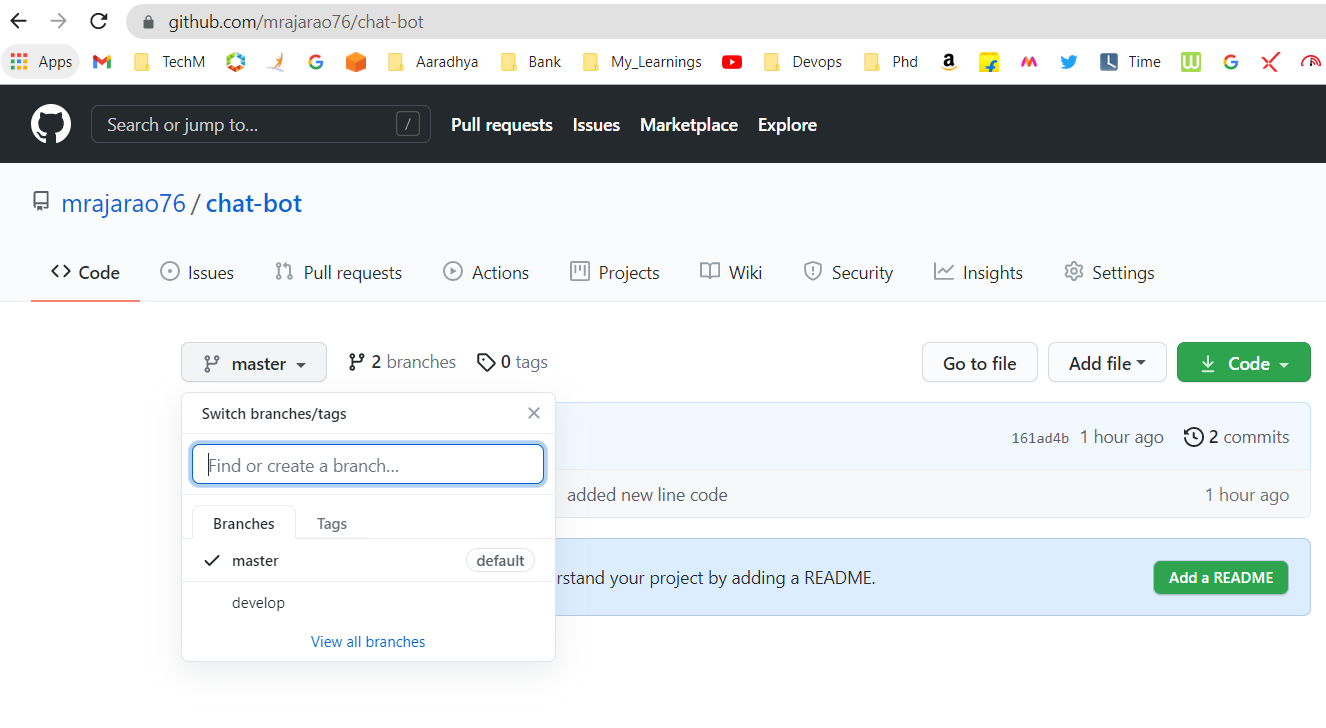
**$ git remote -v**

**$ git push origin develop**



Once you push the code into the central repository you will see two branches

1. Master
2. develop



**Setting Rules int the GIThub**

All push code will go directly into the develop branch we need only the merge code into develop branch to restrict this we will set up the rules by setting the approval process

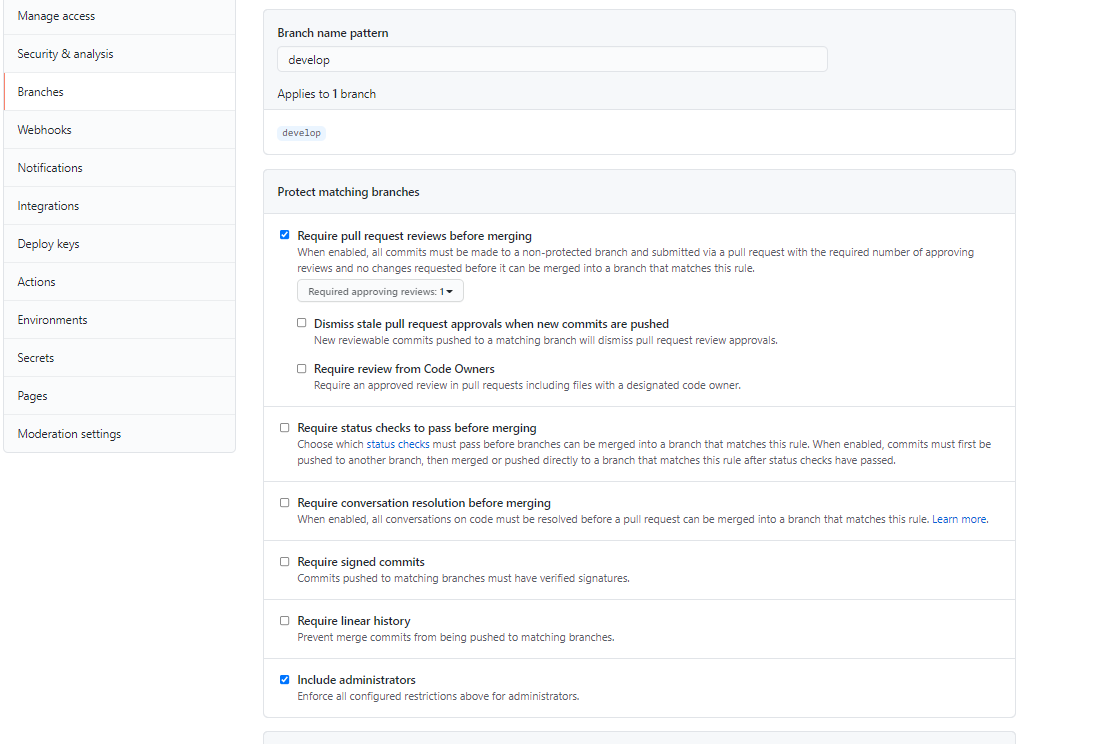
In git hub navigate to settings and click on branch and provide the branch name and add rule and set the following Branch protecting rules

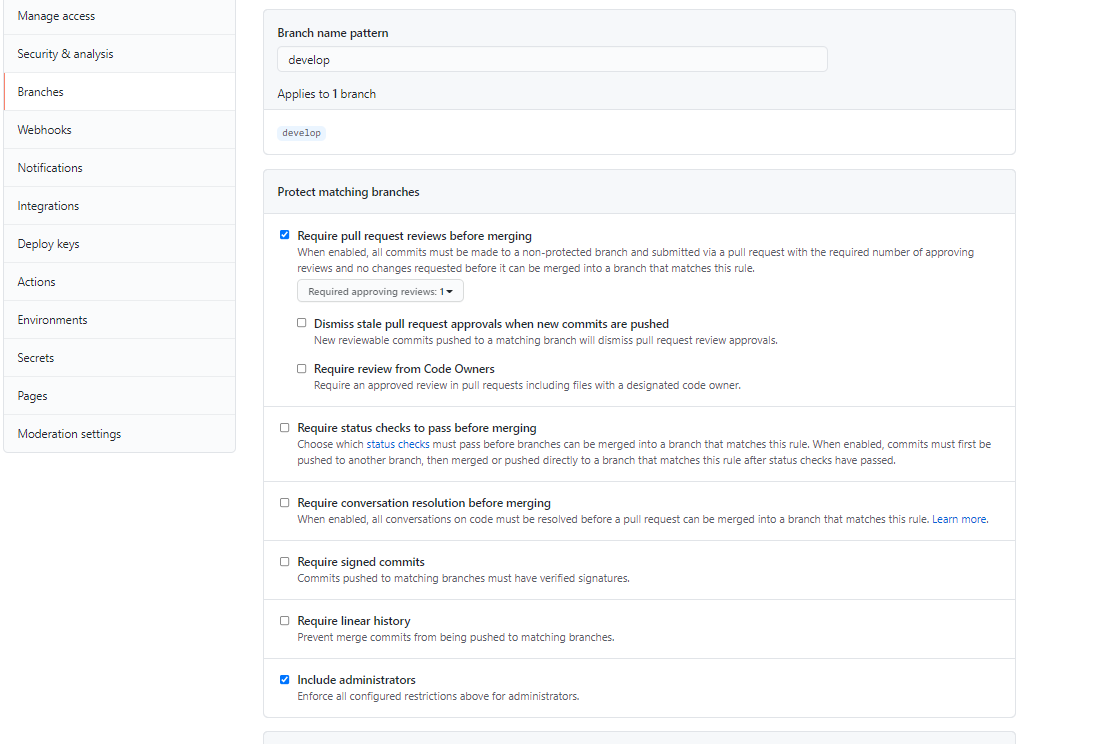
**Require pull request reviews before merging**

When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.

**Include administrators**

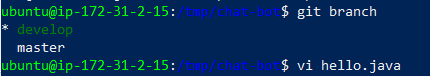
Enforce all configured restrictions above for administrators.





Now to develop branch

**$ git branch**



Change the code in the hello.java file

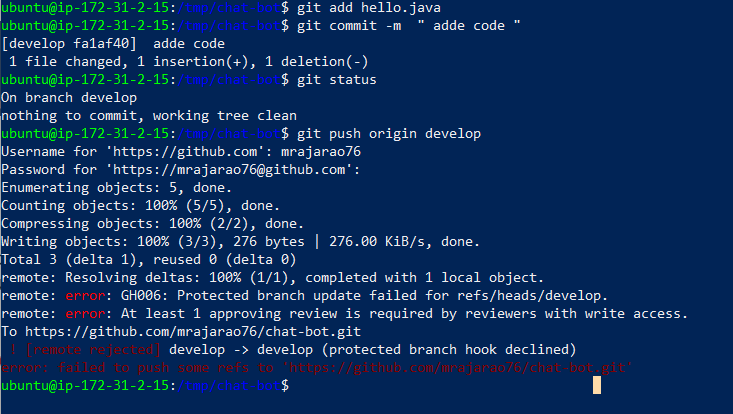
Now follow the step to push the code into the develop which you made the changes it is merging the code as we have set up the rules it will restrict

**$ git add hello.java**

**$ git commit -m “new code added in the develop branch”**

**$ git status**

**$ git push origin develop**



**remote: error: At least 1 approving review is required by reviewers with write access.**

**To https://github.com/mrajarao76/chat-bot.git**

**! [remote rejected] develop -> develop (protected branch hook declined)**

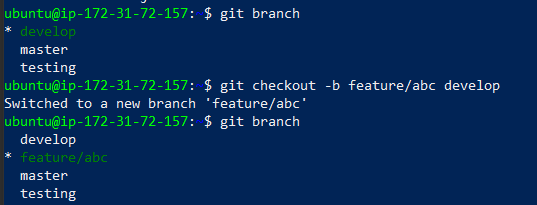
**error: failed to push some refs to 'https://github.com/mrajarao76/chat-bot.git'**

Now creating the feature branch from master.

**$ git branch**

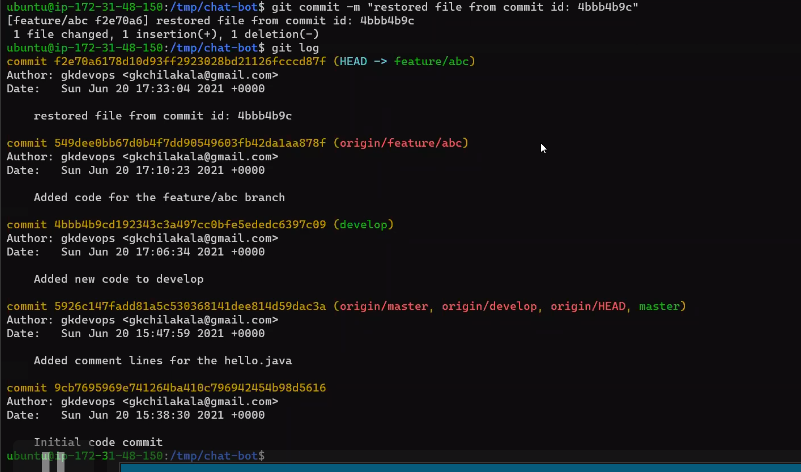
**$ git checkout -b feature/abc develop**

**$ git branch**



To restore the file from commit

**$ git commit -m ”restored file form commit id: 4bb4b9c”**

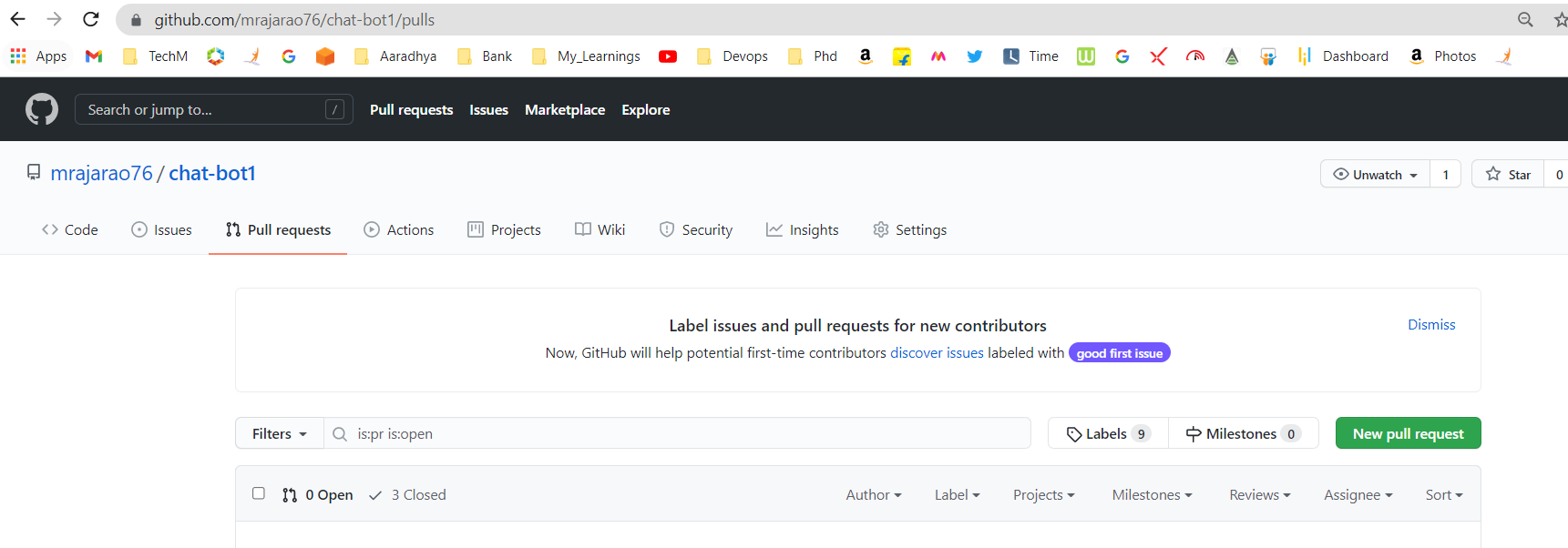


Once you modified/added new code in feature/abc now merge it into the develop branch

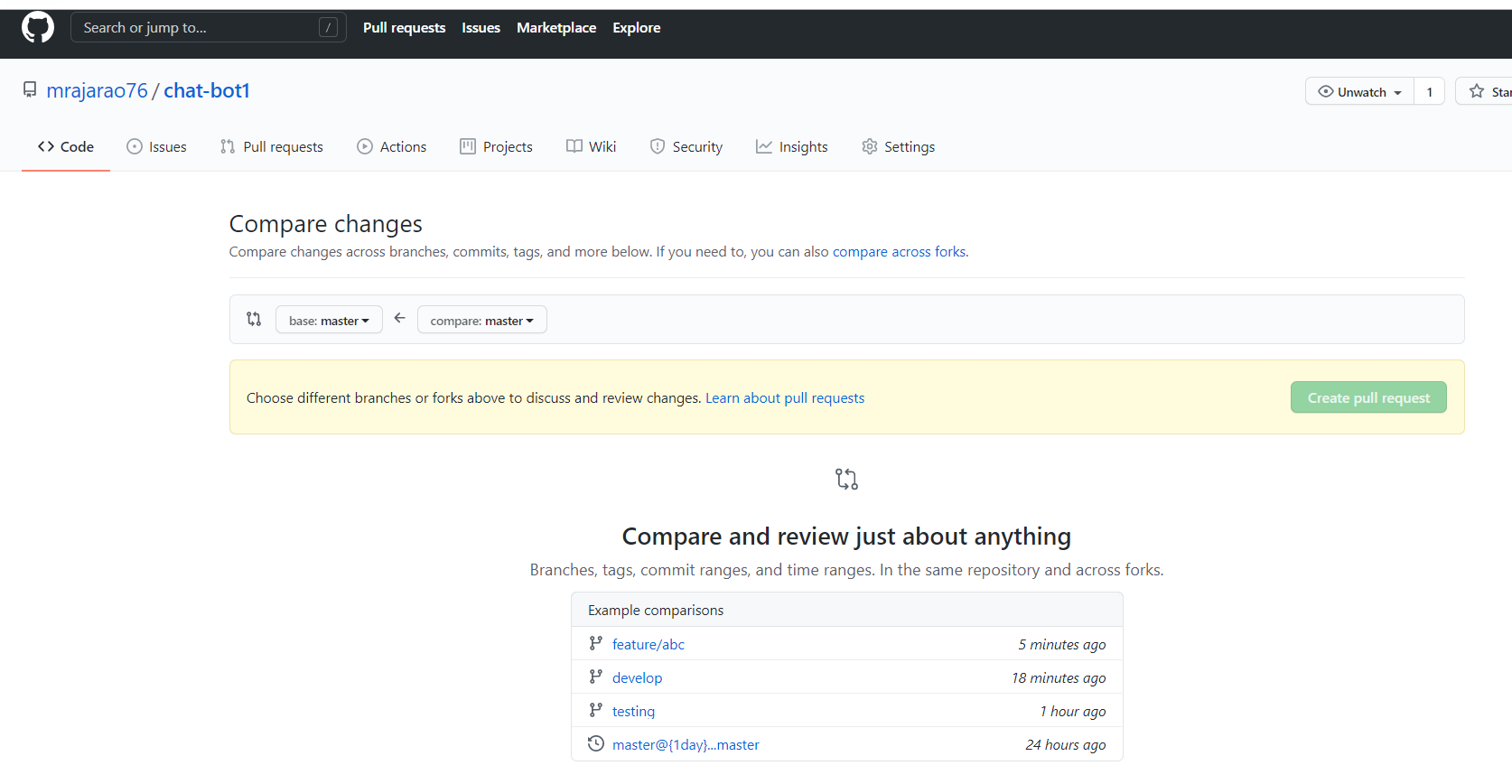
If everything is working fine then we can merge from develop to master branch

Following are the step how to pull and merge from feature/abc branch to develop branch

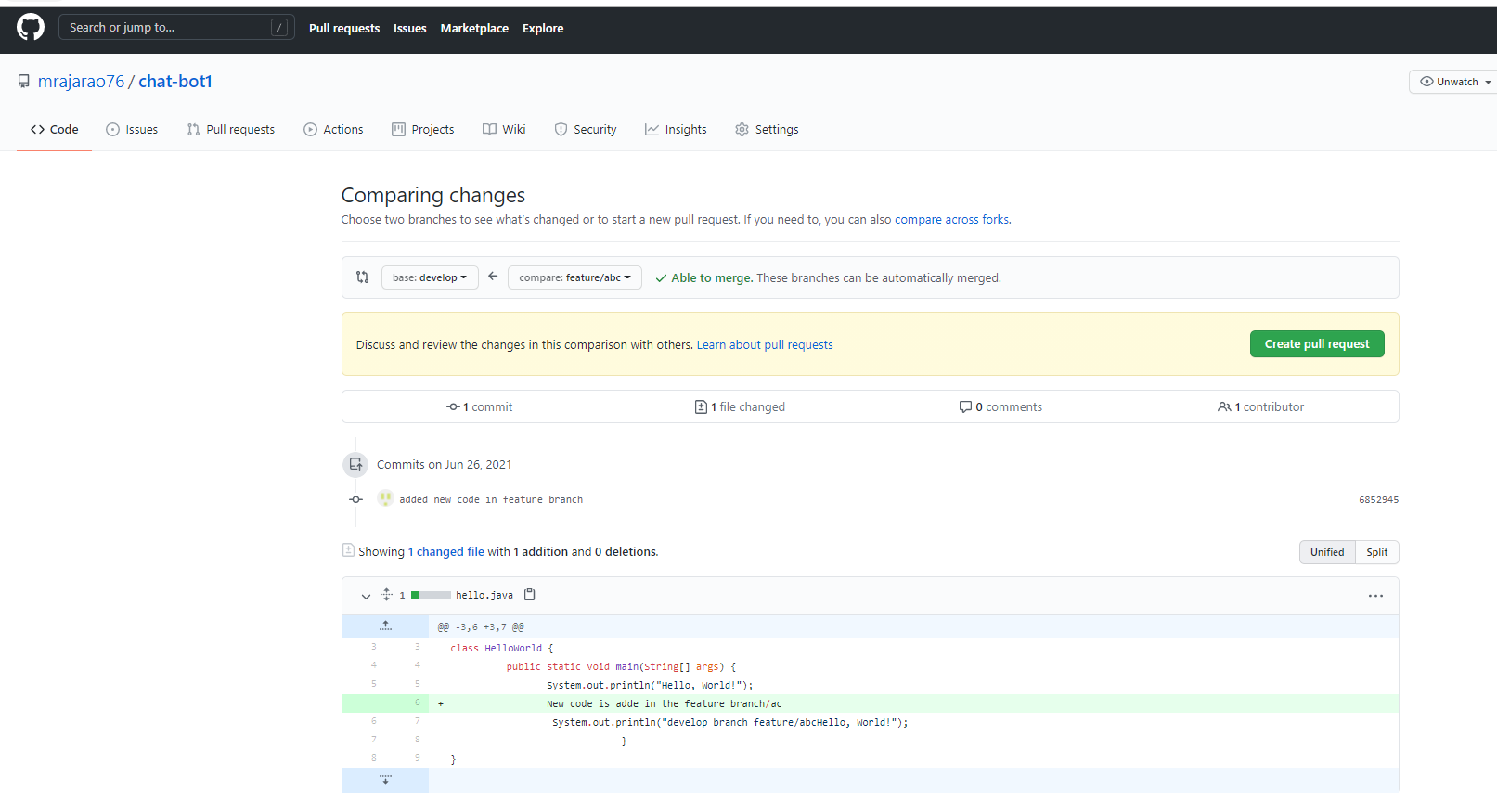
1. Click on the Pull requests



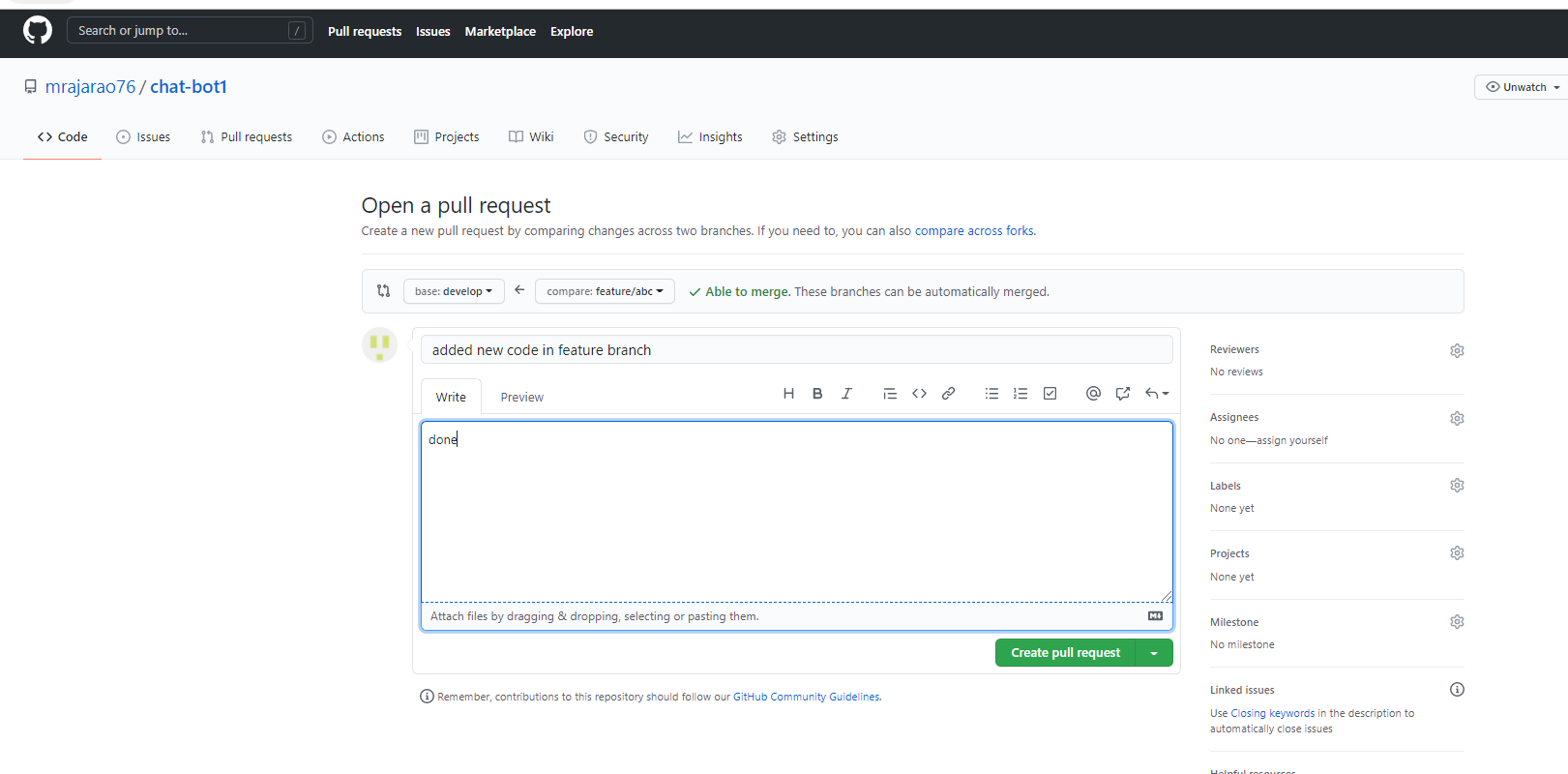
1. Click on New pull requests



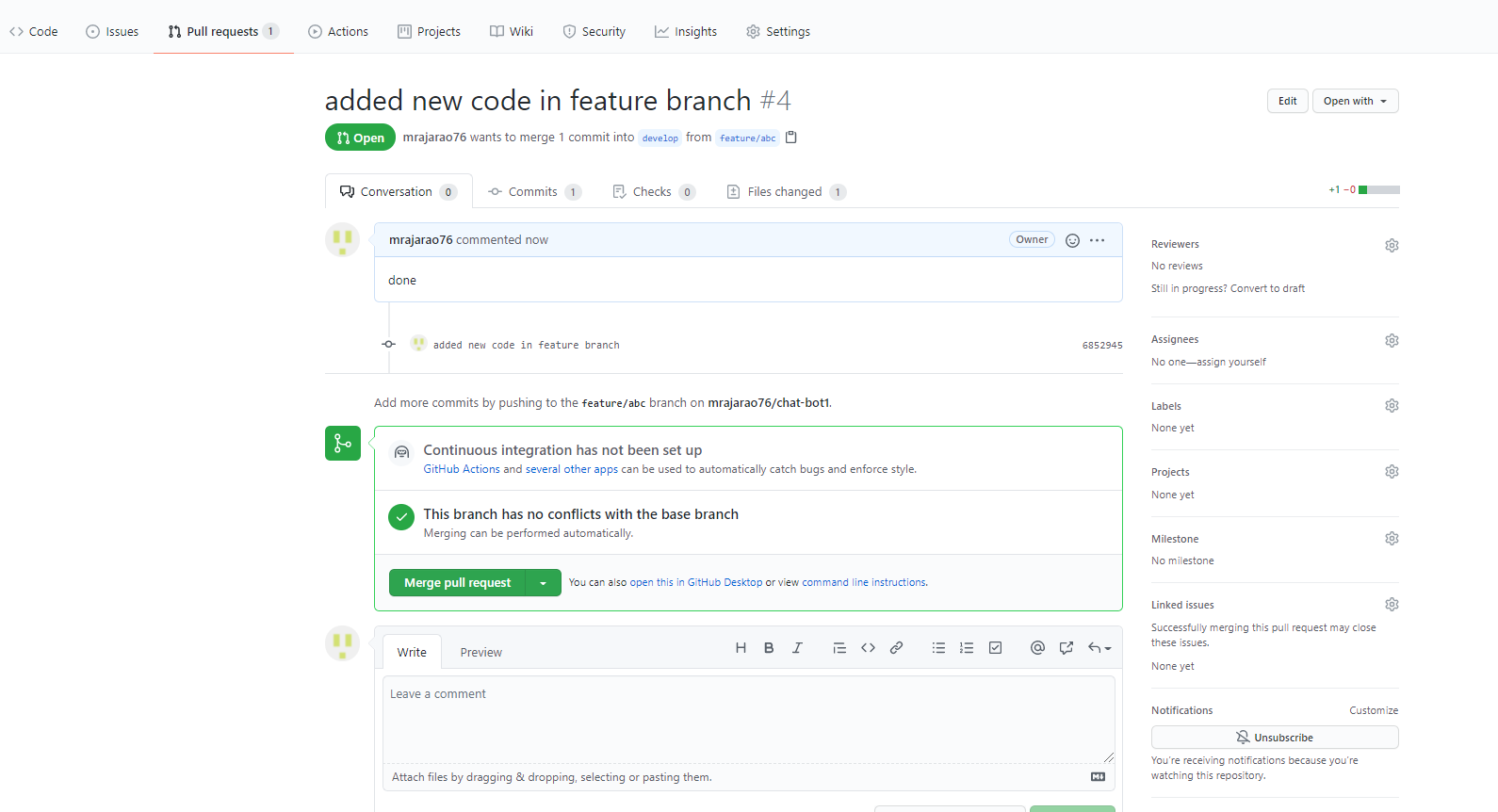
1. Compare the source and destination i.e. base and compare



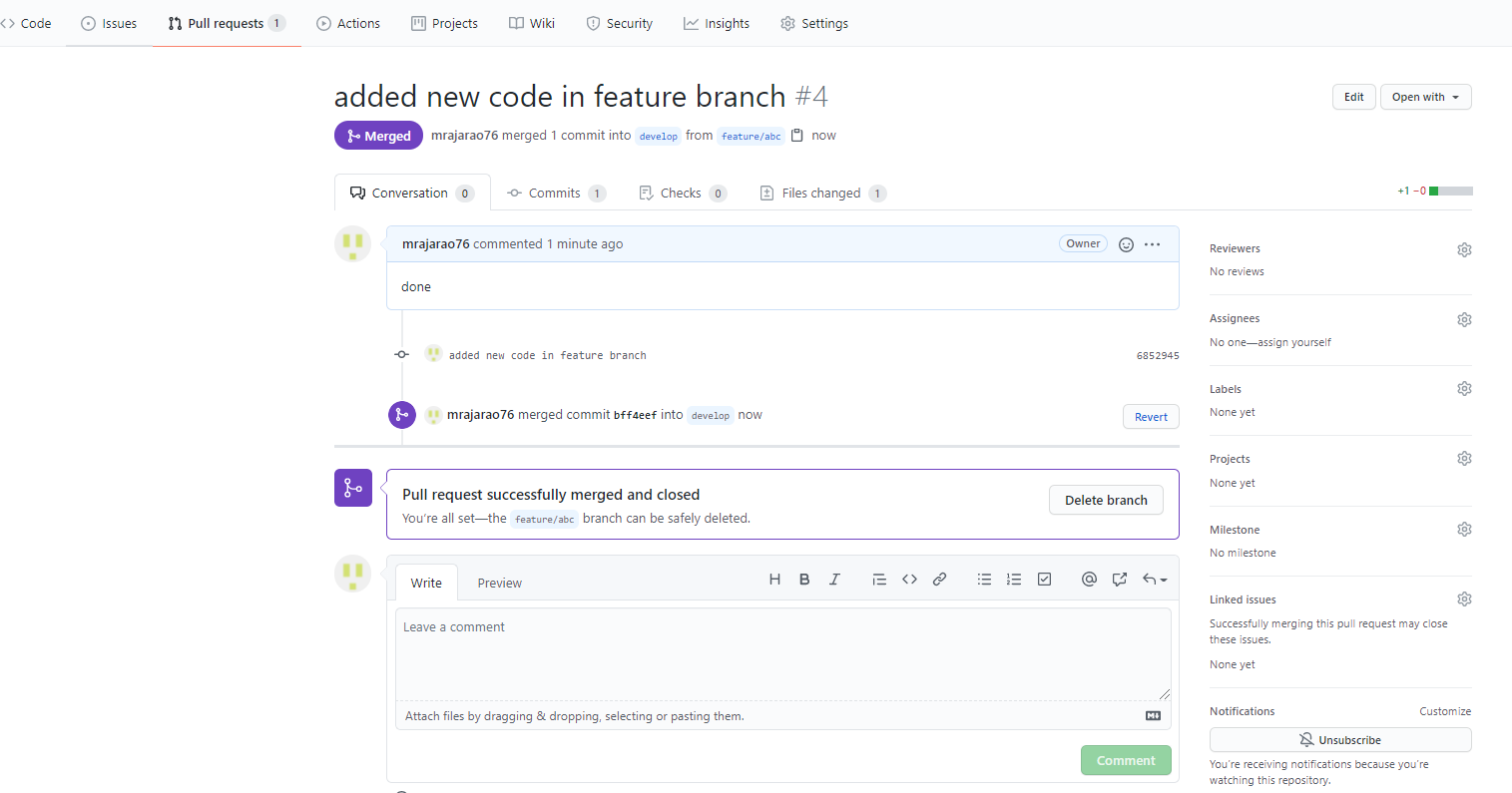
1. Create pull request



1. Again create pull request



1. Click on Merge request and the confirm reques



Following the above steps to pull and merge from develop branch to master branch