URL’s

Git - Reference (git-scm.com)

<https://www.youtube.com/watch?v=xuB1Id2Wxak>

https://www.youtube.com/watch?v=b5oQZdzA37I

https://www.youtube.com/watch?v=GFUikJWaMn0

**GIT:**

**Version Control System**: VCS is one type of Management, It allows you to keep track of your work and helps you to easily explore the changes you have made like computer programs, Changes in documents, source code and large web sites…etc.

These changes are usually called as “versions”.

**For Example:**

Suppose I have developed one web page like mysite.html in 2005 and later I have added one more web page like mysite2.html in 2007 and I have added photo.png in mysite2.html and on 2008 I have modified the page layout of the mysite2.html.

These changes all called as versions.

Suppose if you make any mistakes in previous versions (mysite.html)

We can roll back to the previous versions and do changes if required that is the whole purpose of maintaining the version control systems.

**Why should we use Version Control Systems?**

Suppose imagine a 3 developer’s working on a particular project in same shared folder (Isolation).

One developer made changes in particular application and in the same application other developer made some more changes and they are continuously doing that modifications on the same application differently. At the end when we try collaborating or when you try to merge all at that time, we got a lot of conflicts and you might not know who have done what kind of changes. So version control system provides a shared workspace and it continuously tells you who has made what kind of changes or what has been changed so you’ll always get notified if someone has made changes in your project so with version control system a collaboration is a page between all the developers and you can visualize everyone’s work properly and it will save a lot of time for you because there won’t be much conflicts.

**What Is Repository ?**

Repository is nothing but main folder, every think is associated with a specific project should be kept in a Repo. Repo can have folders within team or separated files.

You will have a local copy (on your computer) and an online copy (on GitHub) of all the files in the repository.

**(i) Local Repository:**

1.Located on your local machine.

2.Resides as a “.git” folder in your local project root.

3.Only admin of the machine can work with repo.

**(ii) Central Repository:**

1.Typically located on remote server.

2.Exclusively consists of “.get” repository folder.

3. Team to share and exchange data.

**BACKUP:**

If any case if your central server crashes, a backup is always available in your local servers.

(Central Server)

Local server1 Local server2 Local Sever 3

**What is GIT ?**

Git is a Distribution version control tool it allows you to perform all kind of operations like to fetch data from the central server and to push all your local files into central server like that.

**What is GitHub ?**

GitHub is a cloud hosting platform for version control collaboration. It allows you to host the your central repository in a remote server.

**Git Commit:**

Git commit you have made changes in file in your local repo. Which is actually version of your project so let’s say that you have made some changes and we have committed those changes what your version control system will create another commit object and this is going to be different version with the changes. So, your commit snapshots are going to contain snapshots of the project which is usually changed.

**Git inti :**

This command basically creates an empty Git repository in your local system or reinitialize an existing one -basically .git directory

**Git Add:**

Git add command adds new or any changes made in existing files in your working directory to the staging area (index). Without git add no git commit would ever do anything.

**Git status:**

It shows the current state of your Git working directory and staging area (index). It provide only information, it won’t modify commits or changes in your local repository.

**Git Pull:**

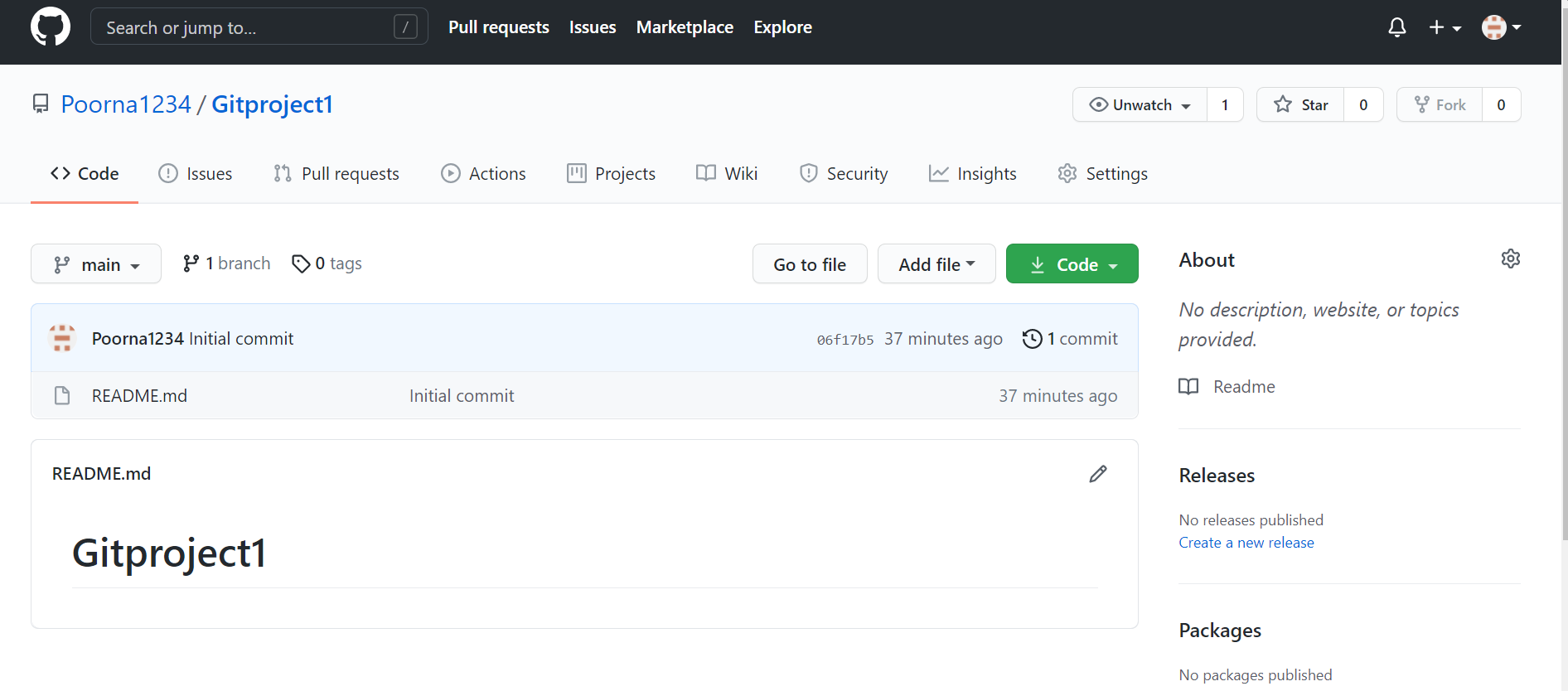
Git pull updates your current local working branch, and all the remote tracking branches. Without git pull your local repository will never be updated with changes from the remote.

**Git Push:**

Git push uploads all your local branch commits to the corresponding remote branch. Suppose if you make any changes in local branch, you can share them with remote repository by using the git push command.

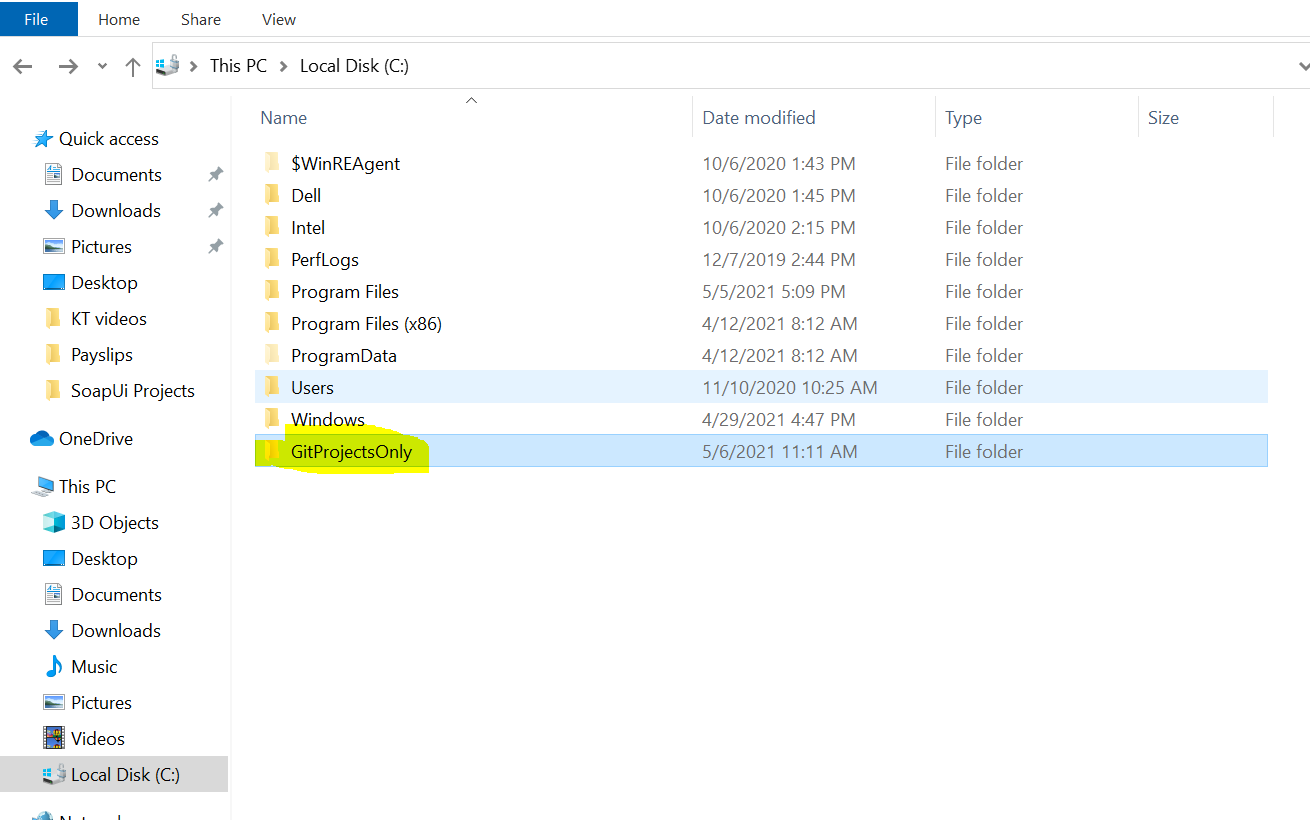
**Hands-On:**

Once central (Remote) repository is created than go to create a local repository in your local system.

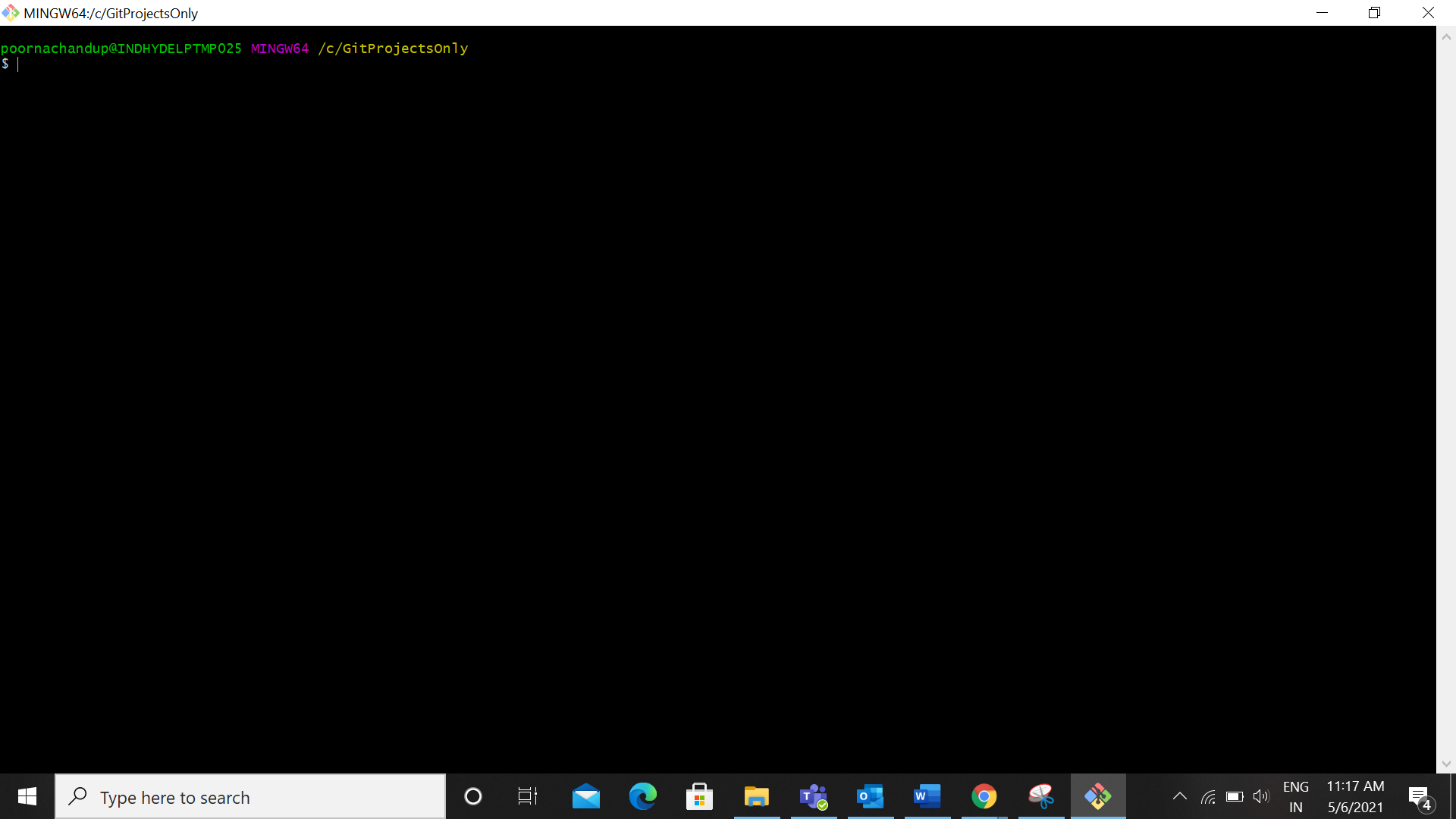


Here main is the branch (master)

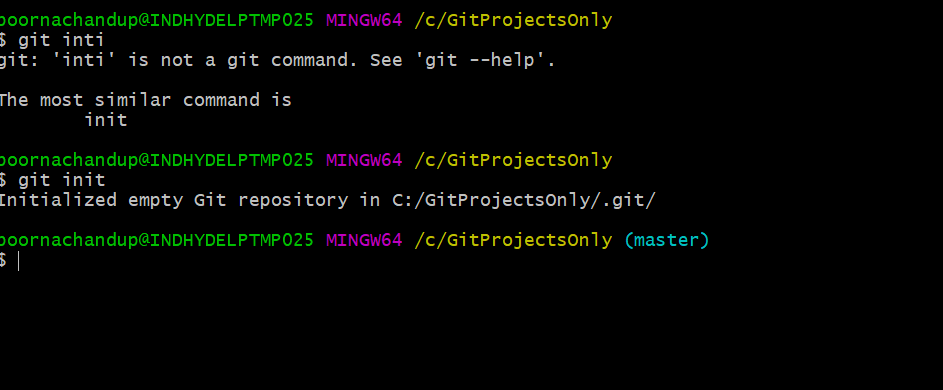
Go to🡪cd🡪create folder



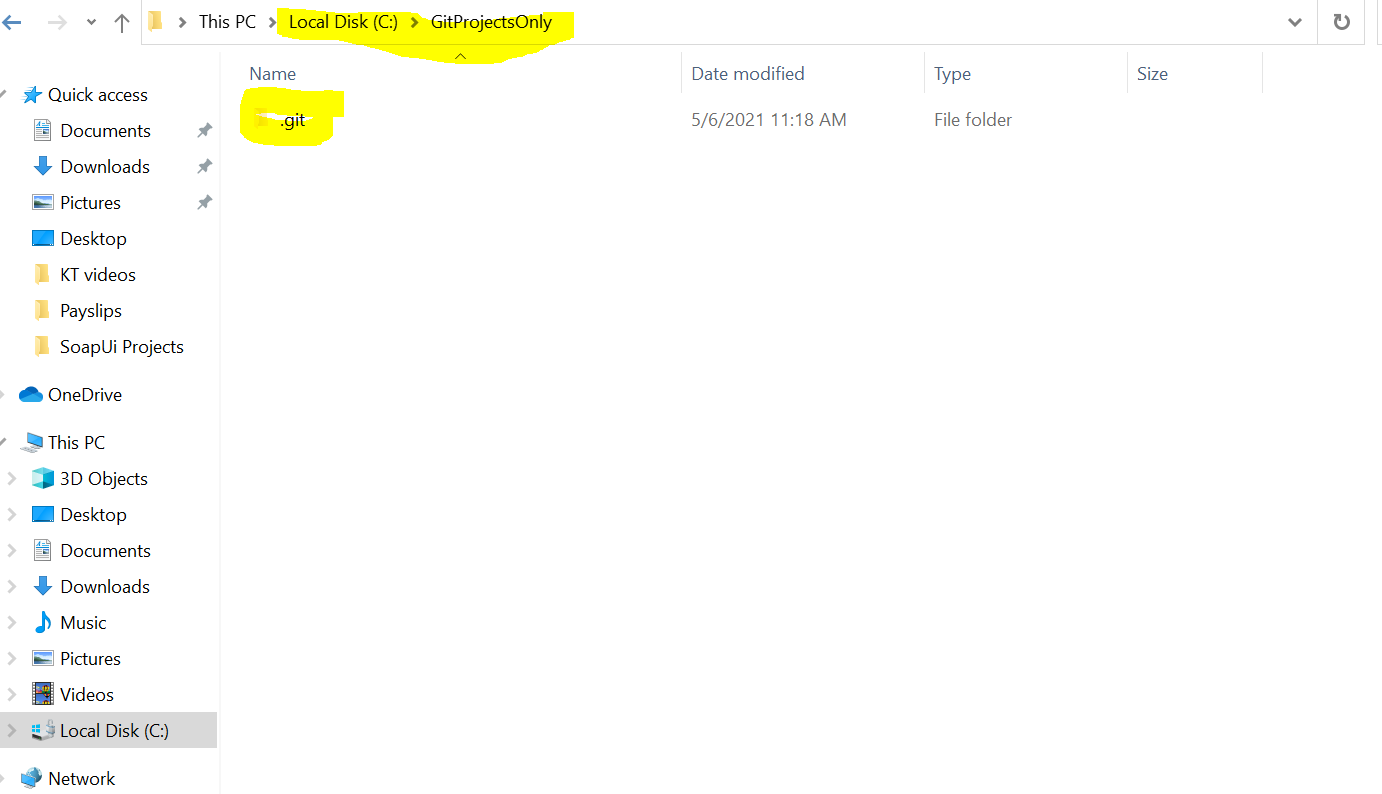
Open the created folder and RC go to git bash and click and got bash emulator



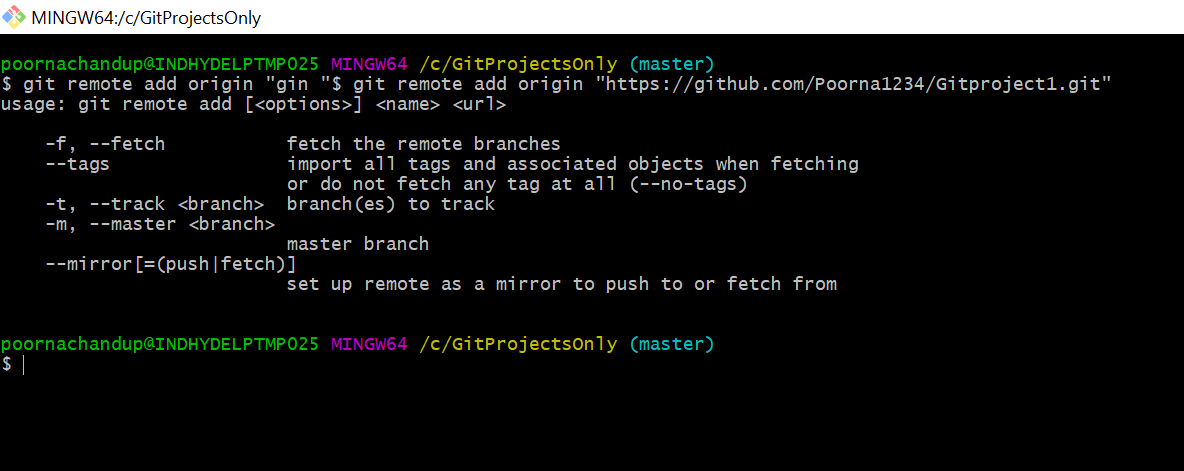
Create a local repository we can write the command like **git inti**



Then go to earlier created folder (GitProjectsOnly) and see .git repository was created

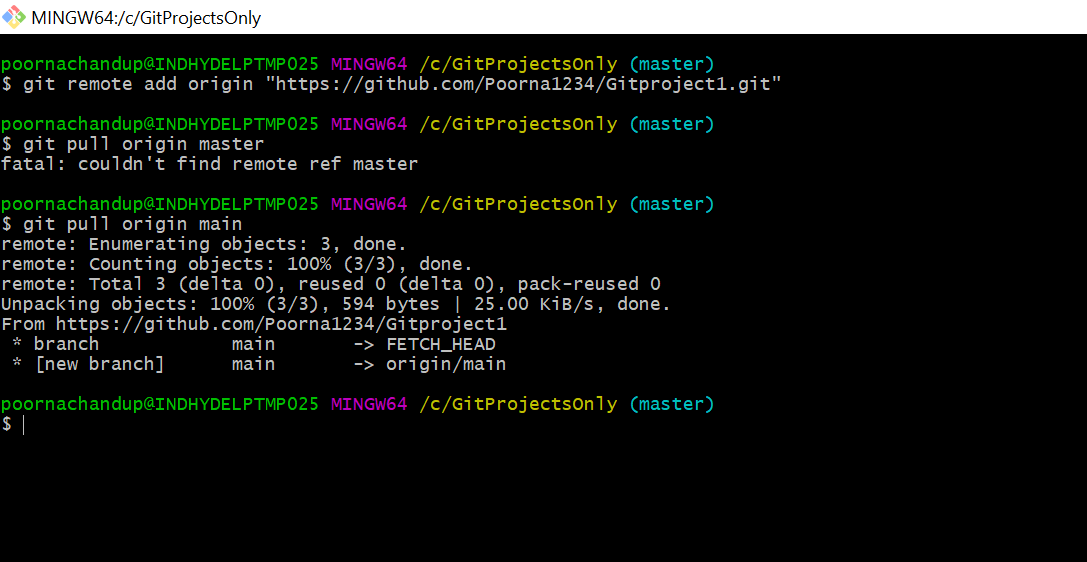


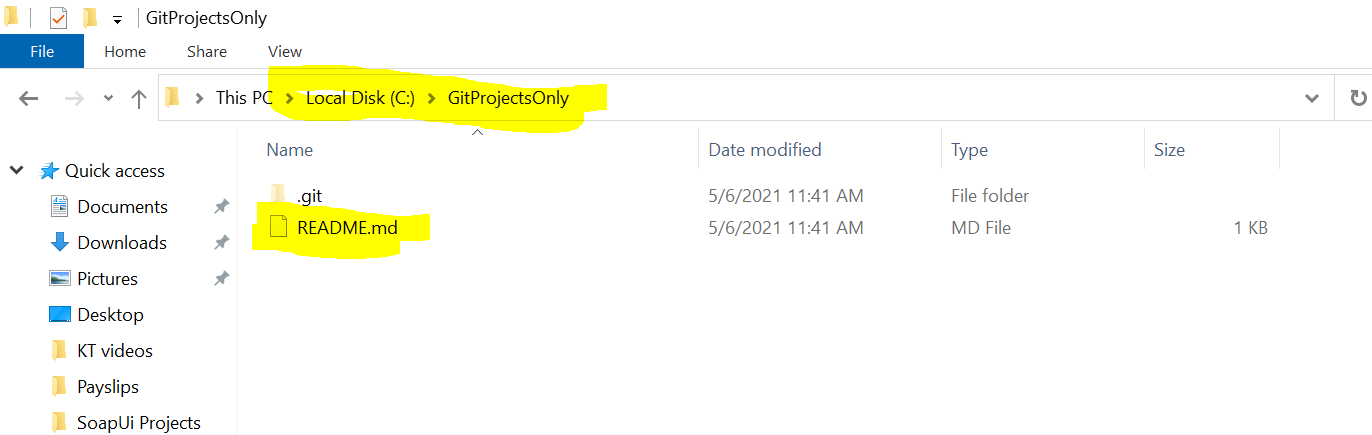
If you want transfer data pull or push from central, we need to link them (link remote repository to local repository) by writing the below command.**(git remote add origin “path”)**



If you want to transfer the data from central repository to local repository write the below command and then check local repository.

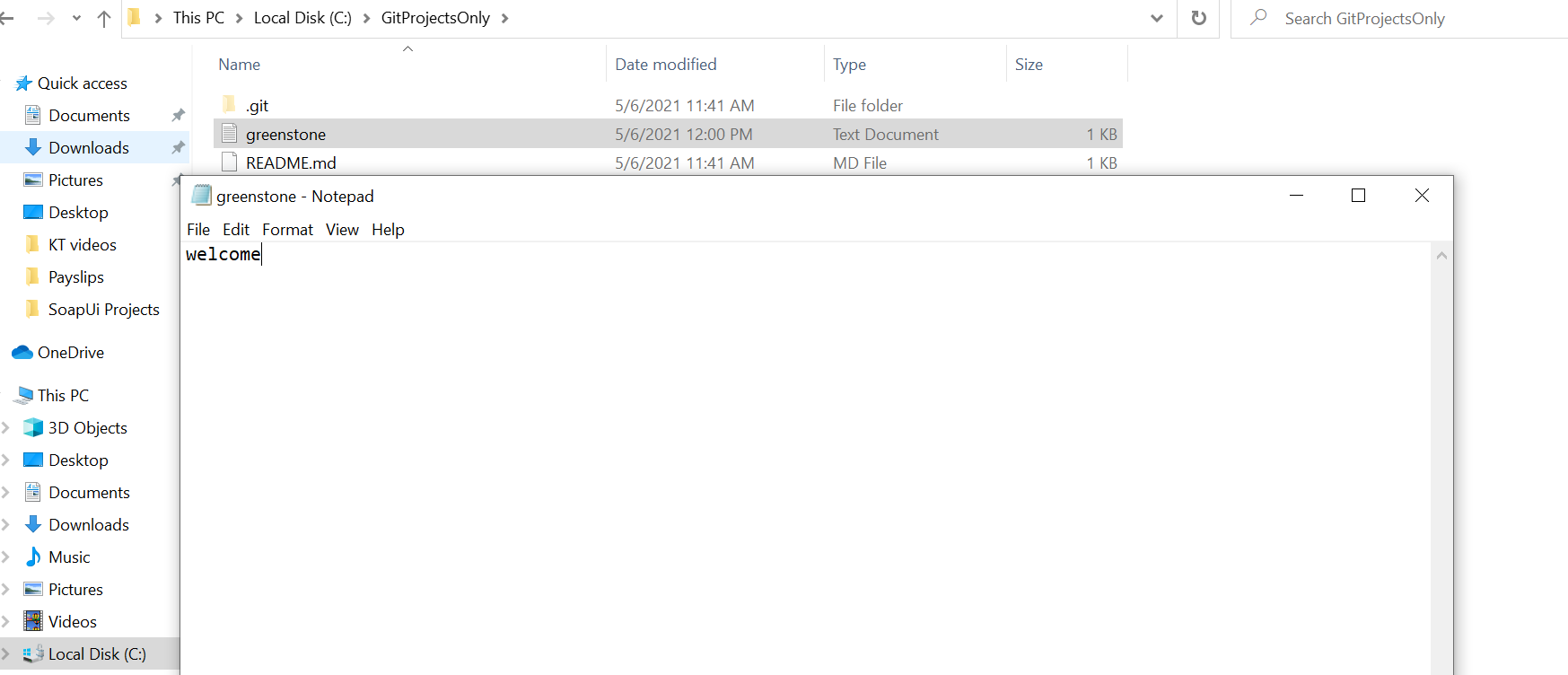
**git pull origin main**



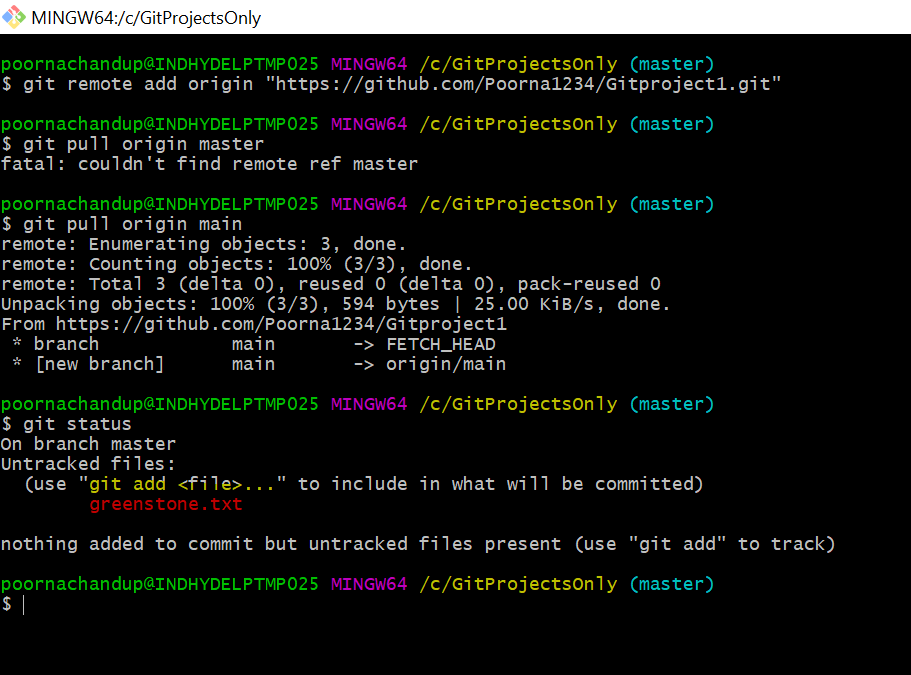


Suppose we make changes in your local repo, we need add and commit those in remote repository

For example, I’m adding a txt file in my local repo

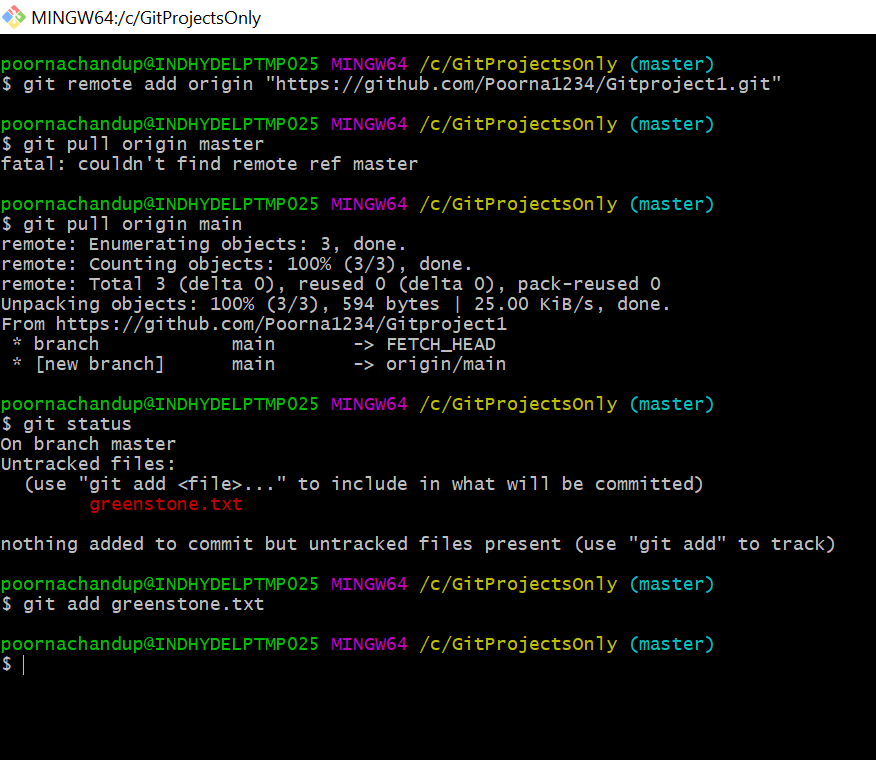


Go to emulator write the command **git status**

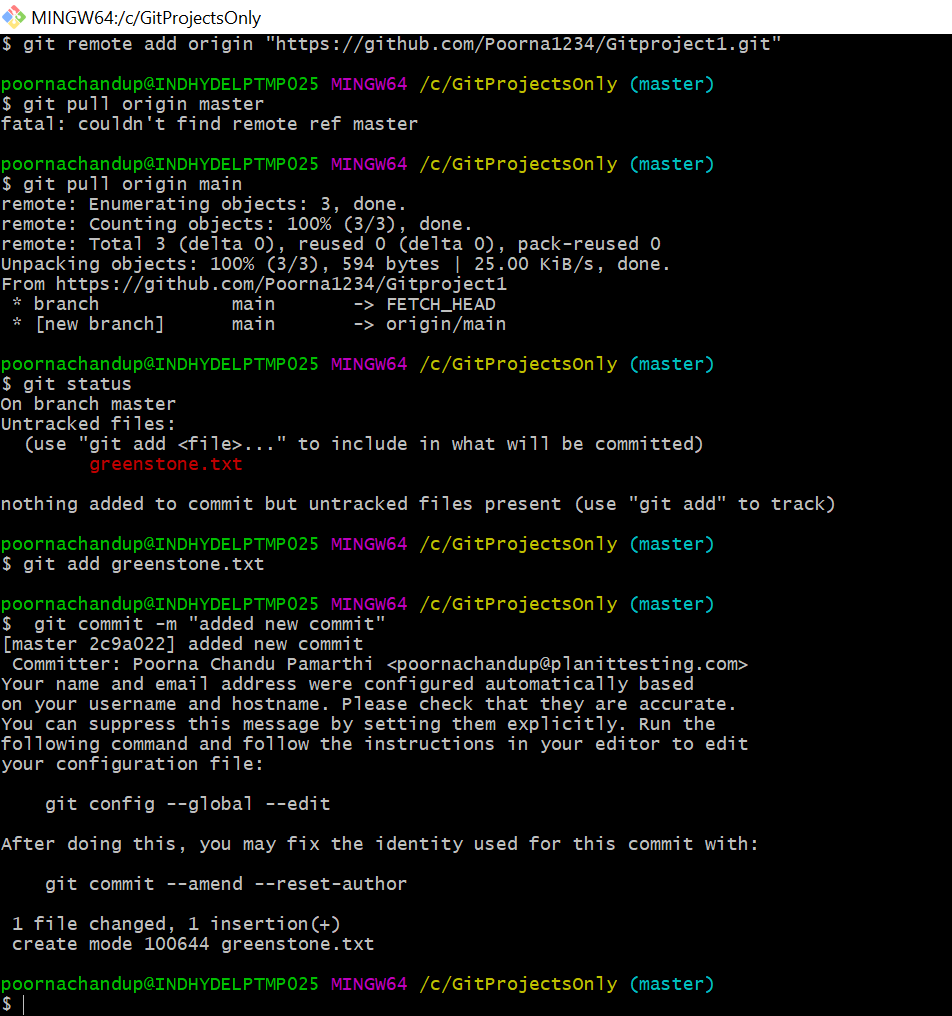


Here we can see greenstone is in red color because it was untracked not in index.

Write the command **git add greenstone.txt**



It was in green now then commit the changes by writing the command **git commit -m “added new commit”**



Above screenshot shows you one file changed, now I’m created two more file with name greenstone2 and greenstone3 in my local repo so now see status



See above txt files are in red and they are in untracked(not in index) so we need to add that files to index and commit the changes.

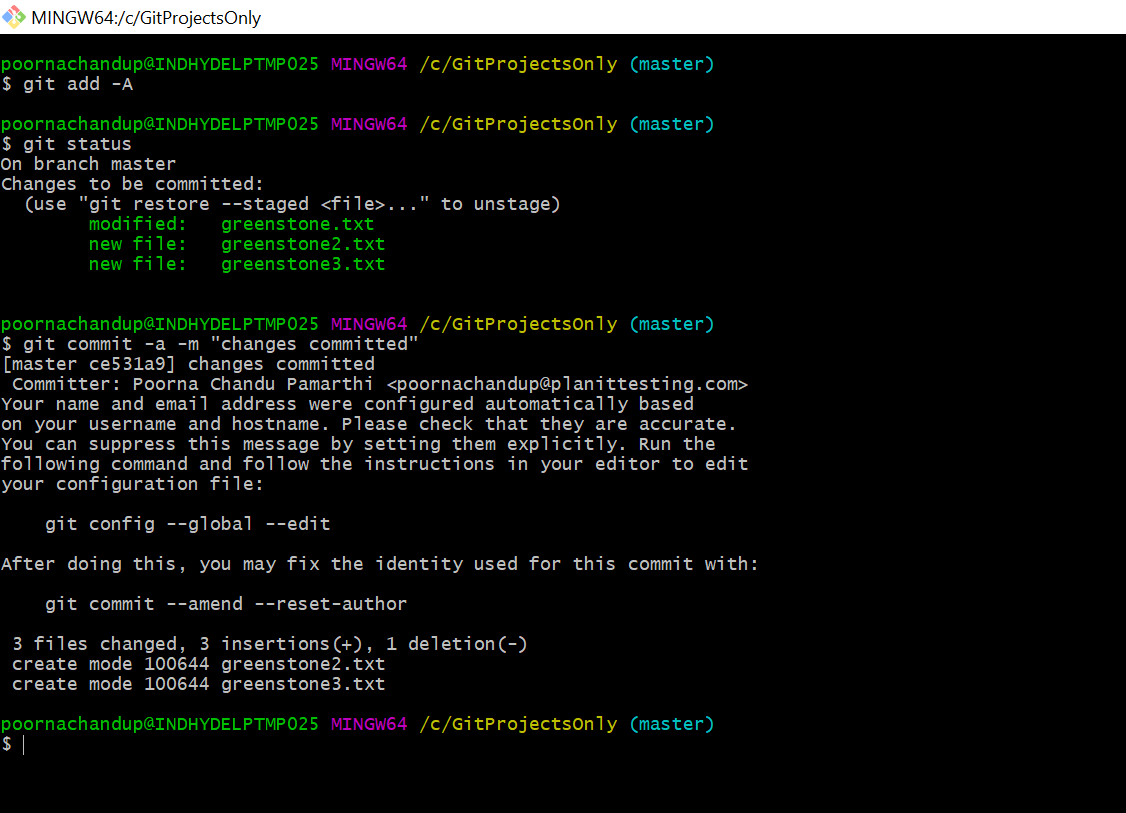
Before that I have modified the greenstone.txt file and see the status.



See they are in red color and modified greenstone file is in index but not committed so it is showing red color.so need to add the files to index write command **git add -A**

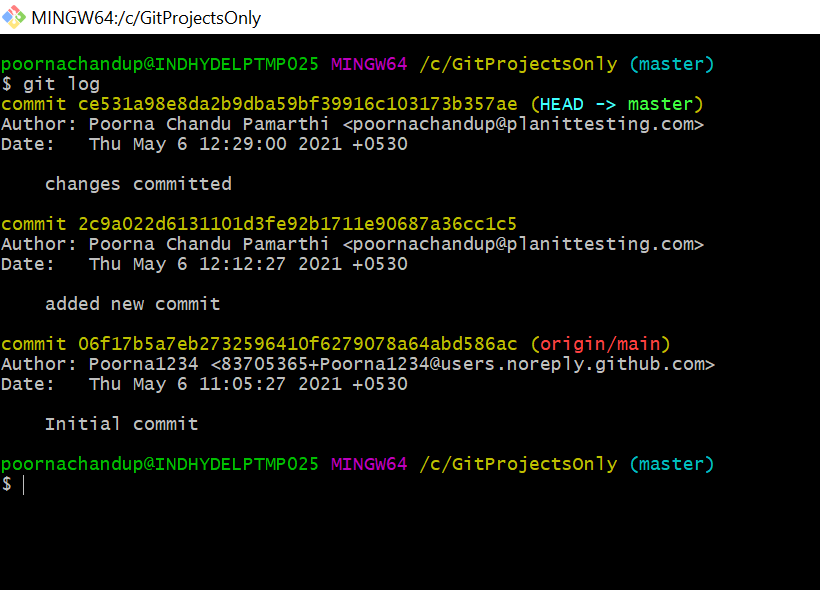


So, if we commit the multiple files, we write the command **git commit -a -m “changes committed”**



Here we can see 3 file have been changes.

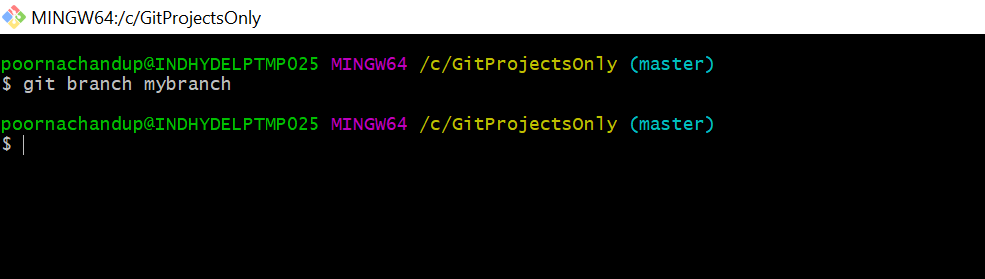
We can write the command like **git log** it will show committed changes of the repository.



Suppose we you add any new features individually in your project or you don’t want to interfere with the master branch- local branches

Remote tracking branches will connect the local branches to central branch.

If you want to create any branch write the command **git branch mybranch**



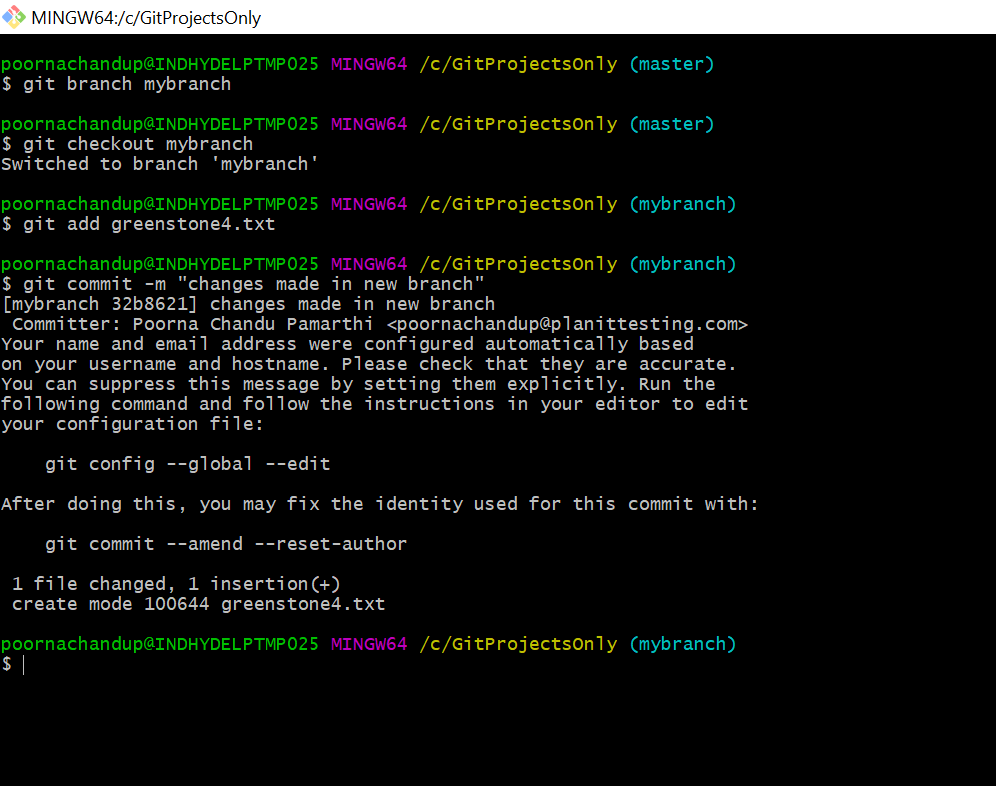
Here new branch is created and it contains the all files from master branch because it is originated from the master branch.

Suppose if you want to move the one branch to another branch, we need to write the command like

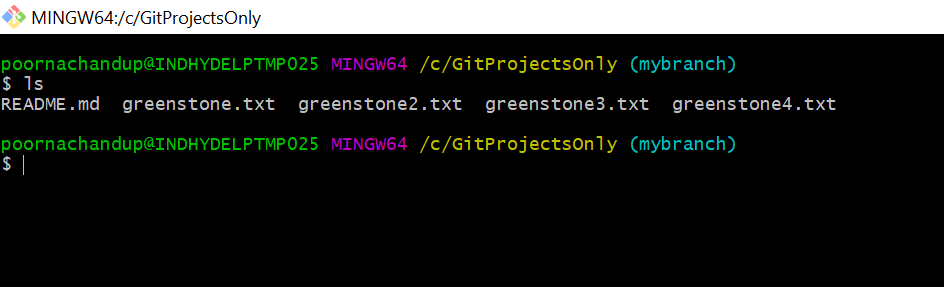
**git checkout mybranch**



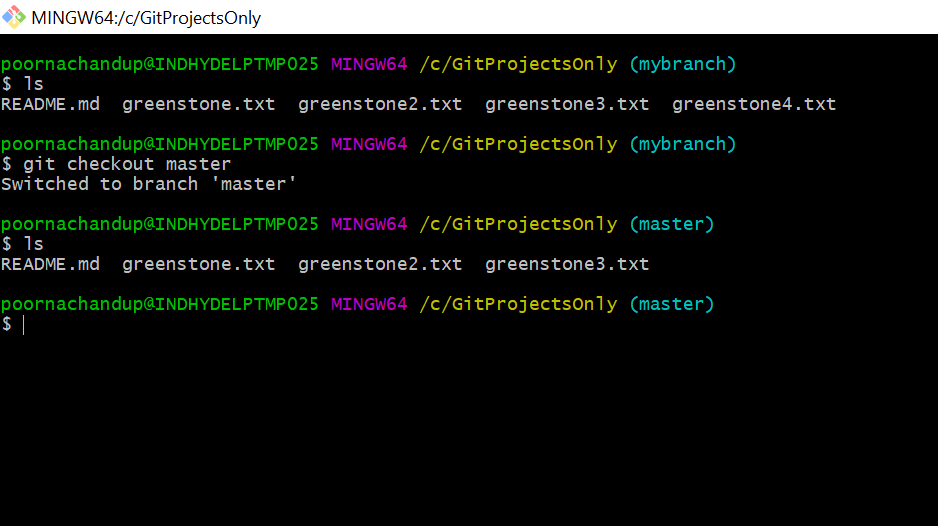
So, I’m now in my new branch so I have created one more files(greenstone4.txt) in my local repo. And add that file in index and commit the changes refer the below screenshot



List out the available files in the new branch you can write the command **ls**



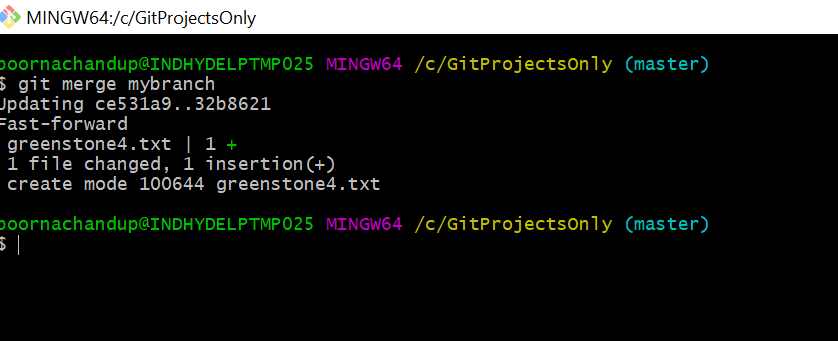
now switch the master branch and check the available files in it.



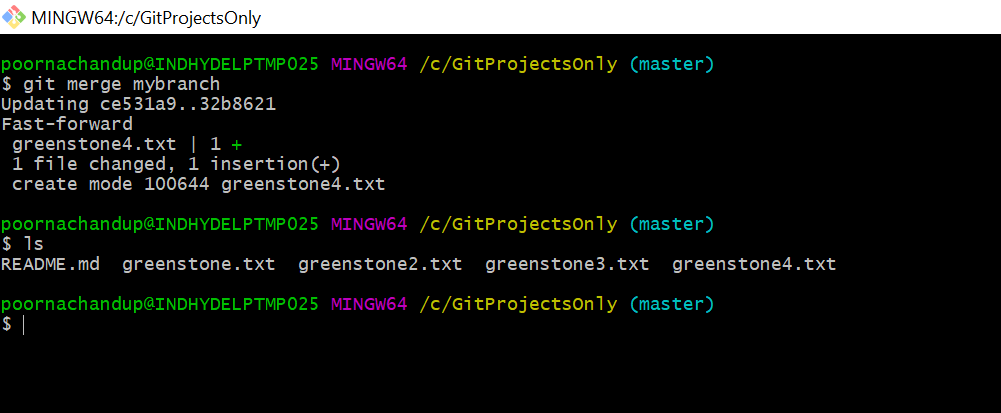
Here you see master branch does not have the greenstone4 file.

Suppose any new files added or changes you need to merge that files back in to master branch.

For example, here greenstone4 related to new branch (mybranch) you need to merge them into master branch so we use command like **git merge mybranch**



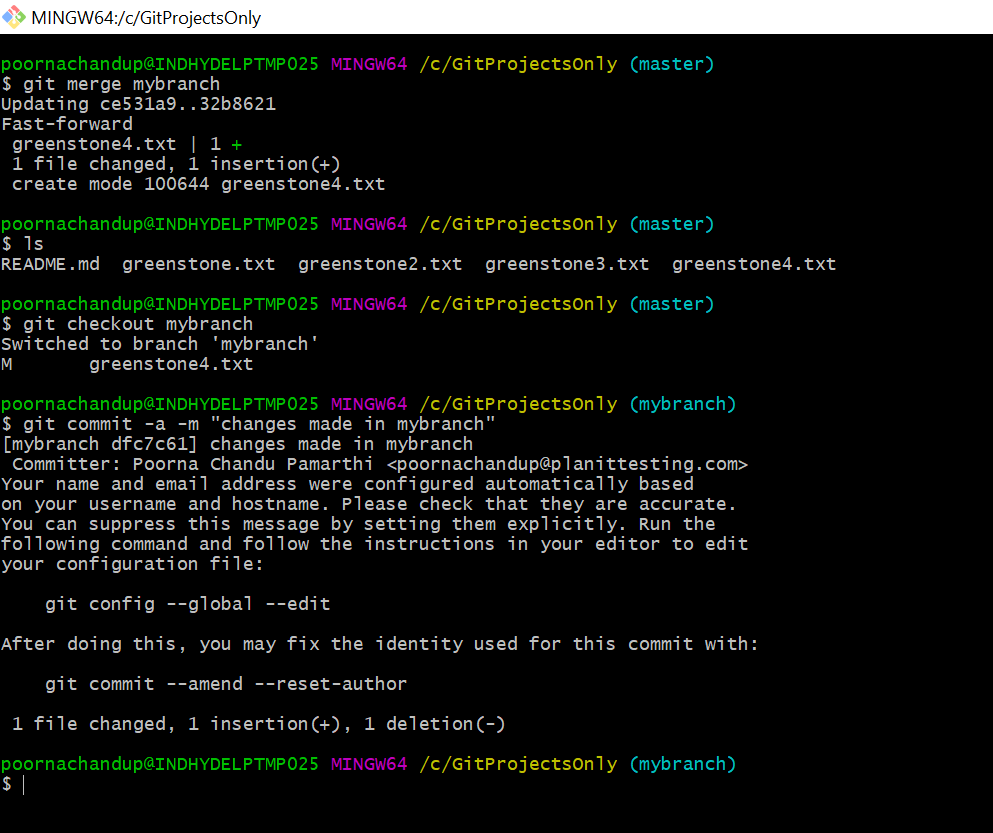
Here you see our destination is master and 1 file have been changed and now you can see the available file after merge

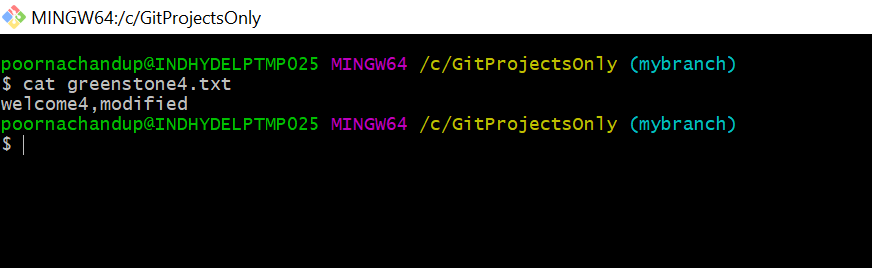


Successfully merged and greenstone4 file available in master now

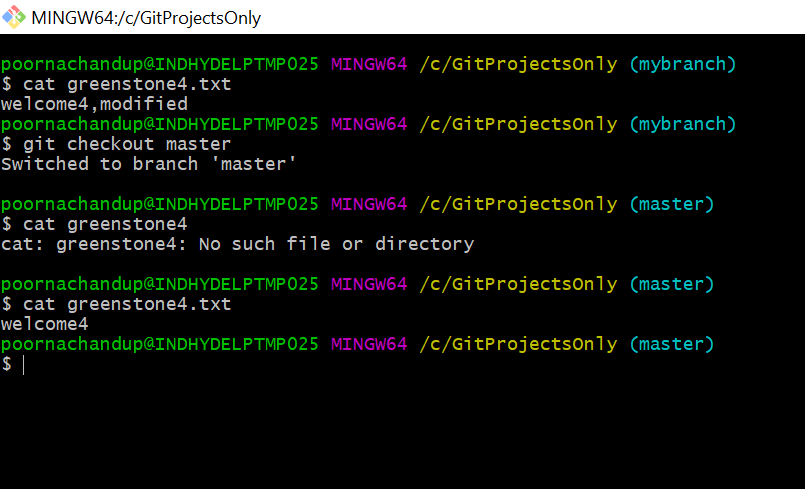
If you made any changes in the new branch files it won’t affect the master until you merge it

Suppose I have done some changes in the greenstone4 file and it won’t reflect on the master refer the below screenshot.





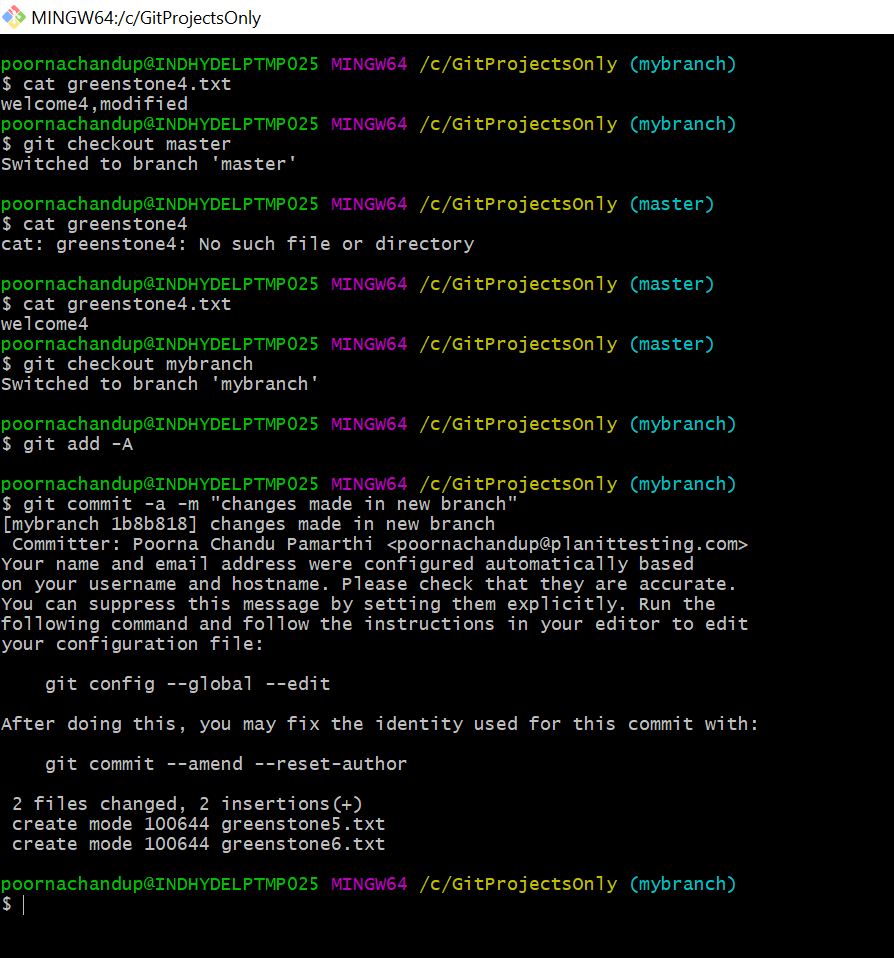
Here changes should visible in mybranch now check the master branch



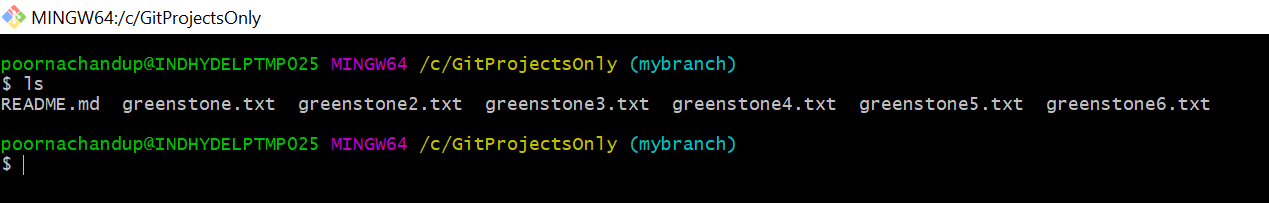
Here you see in master branch it visible only welcome4 not visible modified data we need to merge it.

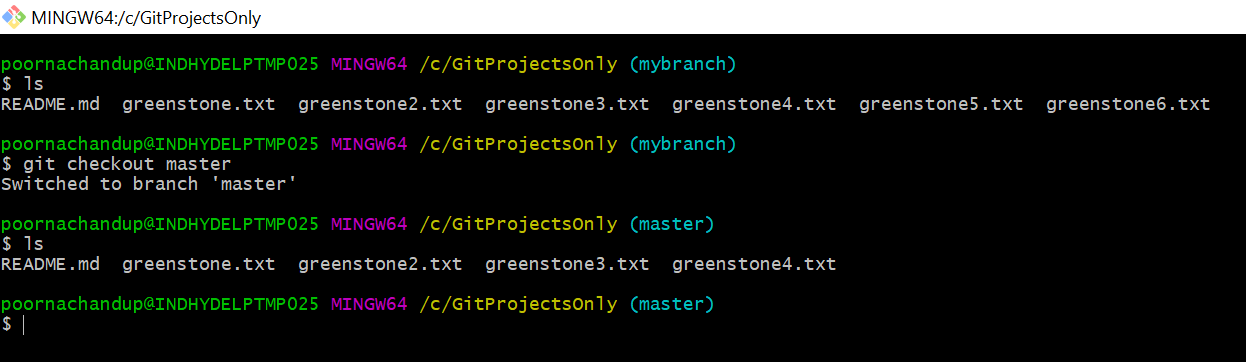
So, switch to new branch (mybranch) and add few more files like greenstone5 and greenstone6.

You need to add and commit those files because they are not in index. refer below screenshots

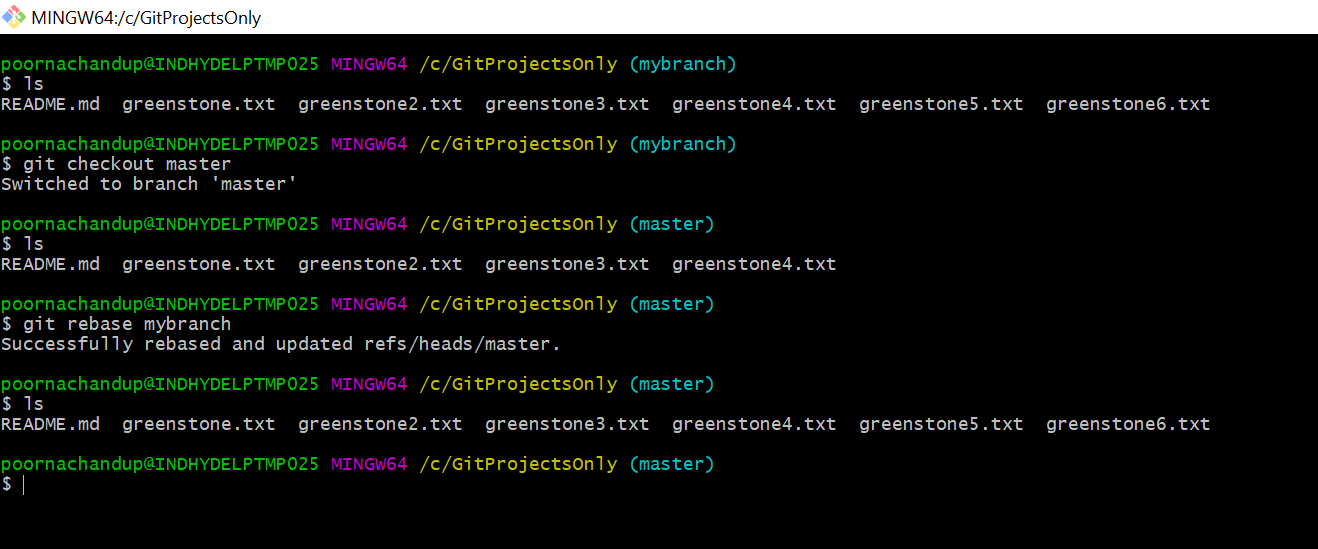


Now you can see the available files in the both the branches

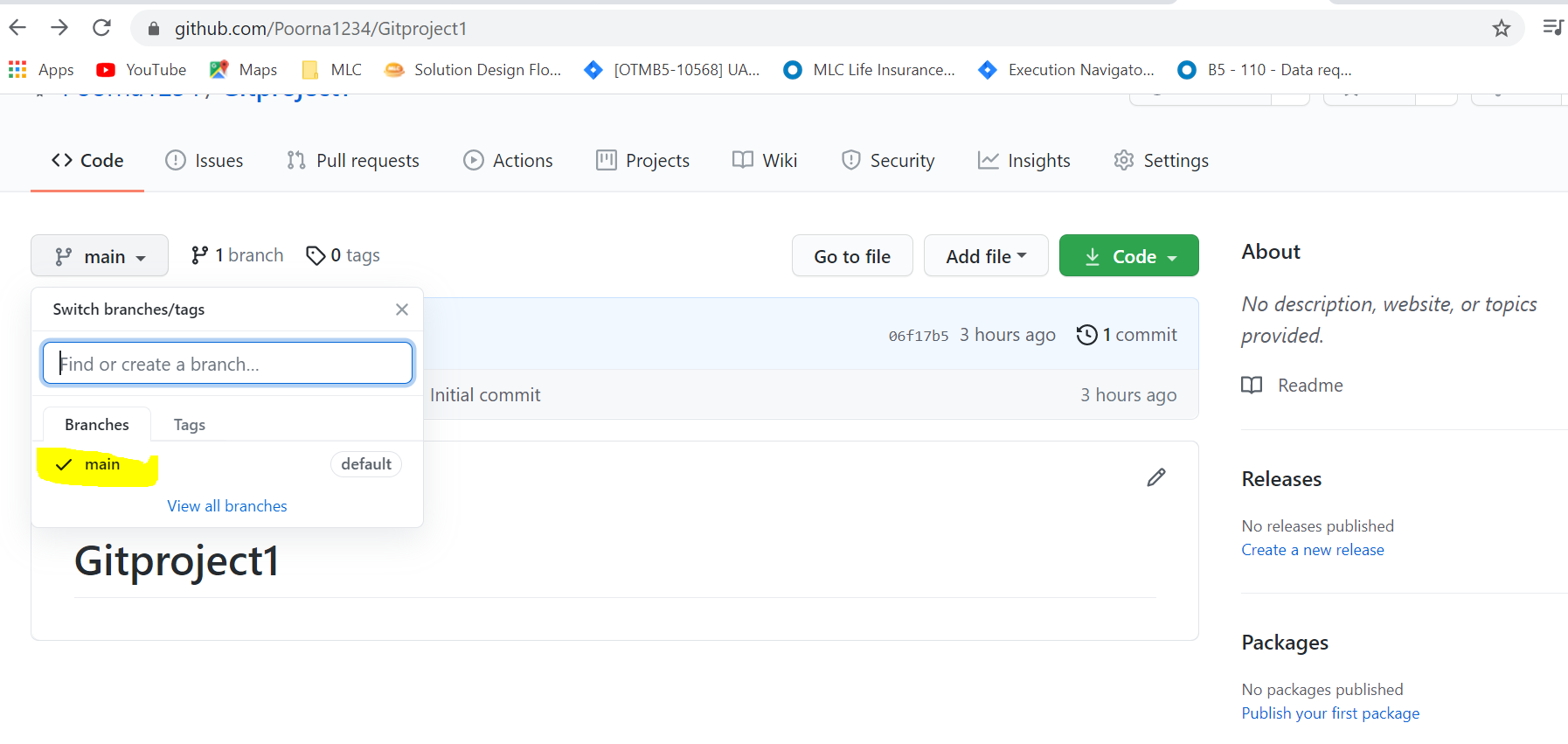




So here instead of git merge I’m using the rebase, it was also doing the same like merge write the command like **git rebase mybranch** and check the available files after rebase.



Now you push branches to central repo, currently center repo having only one branch refer below screenshot



So write the command like **git push origin mybranch**

