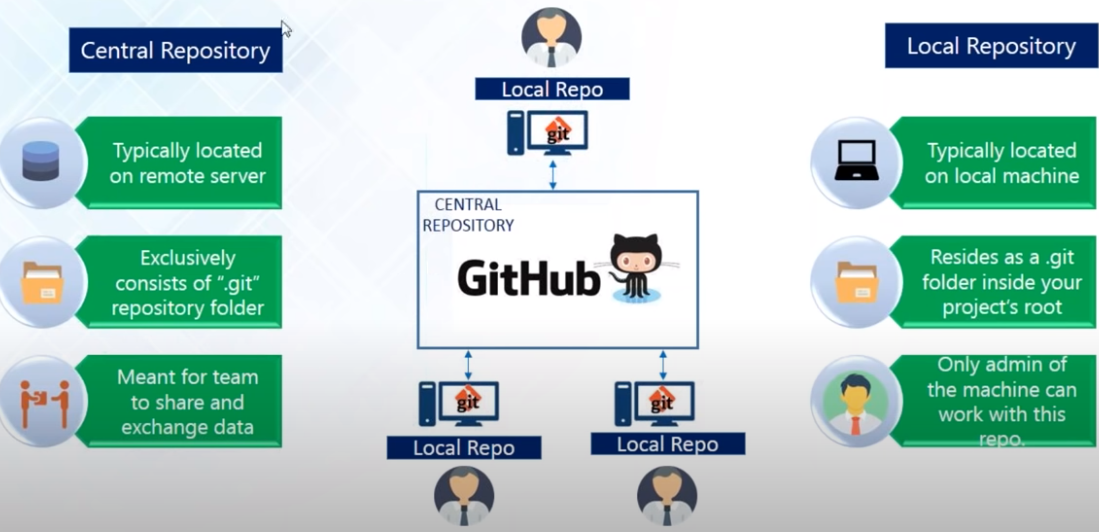
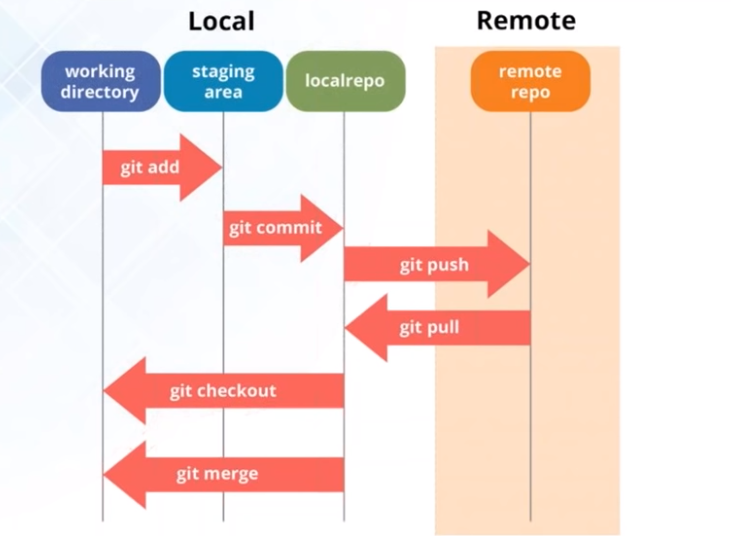
GIT

GIT is a distributed Version tool that supports distributed non-linear workflows by providing data assurance for developing the quality software.

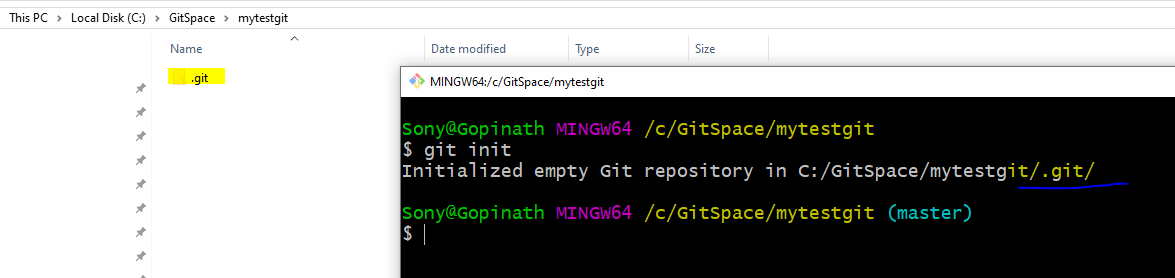
1. Distributed
2. Compatible – can easy to migrate from other version control system
3. Non-linear – a tree and index based structure to navigate and visualize the development history.
4. Branching – GIT is the only one have this feature which takes few seconds to create and merge branches.
5. Lightweight – uses compression technique to compress the data on the client machine.
6. Speed – Faster fetching data from local repository.
7. Open source
8. Reliable – If system crashes. Data can be recovered from any local repository.
9. Secure – uses SHA1 crypto algorithm for every commit and and checkout.
10. Economic
11. Central Repository
12. Local Repository





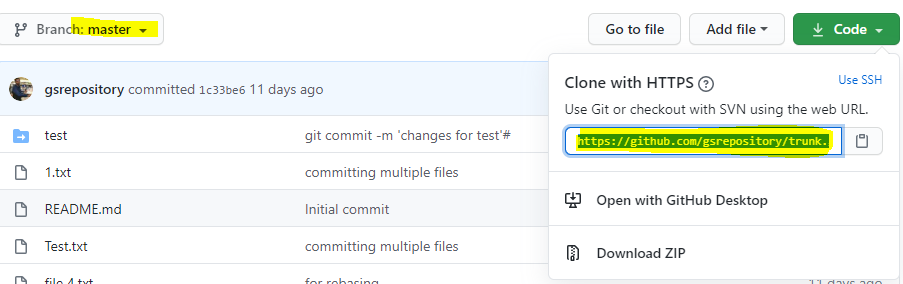
1. Creating Repository
2. Right click and go to Git Bash

git init – command creates the current folder and a **.git** folder is created and out repository is created.



1. Now connect with the Remote Repository to which you want to push or pull.

Go to the github repository and copy the URL as shown below

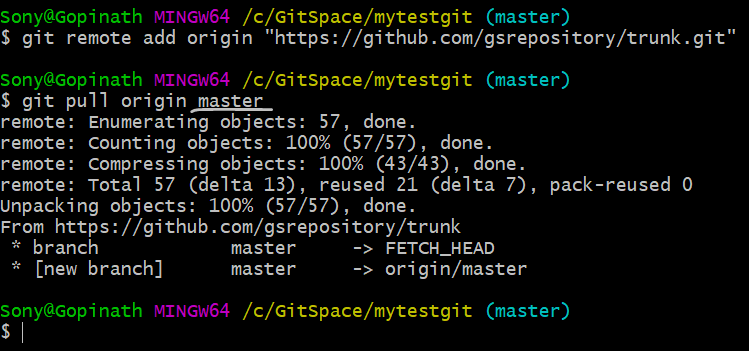


Now from GitBash,

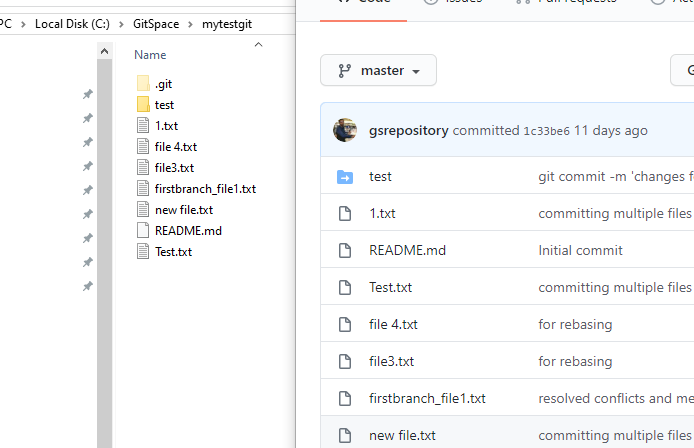
git remote add origin “https://github.com/gsrepository/trunk.git”

and then

git pull origin master – pulls/checkout the source from the master(default) branch



After pull command, the source is pulled/checkout to your local repository, check your local folder should be same as the github repo as shown below



1. Making the Changes

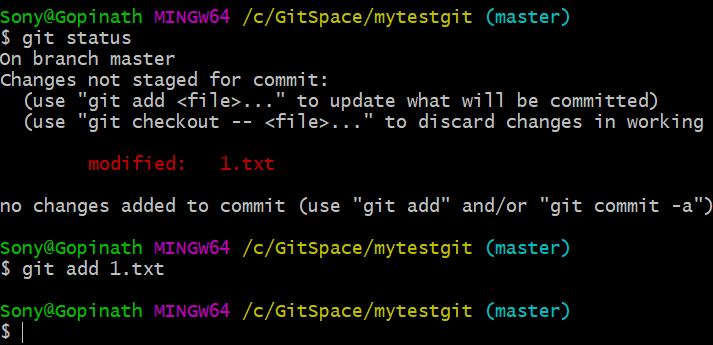
Git commit – saving the changes to repository.

Git add – adds the files to the index

Git status – shows the filed are added to index and ready to commit

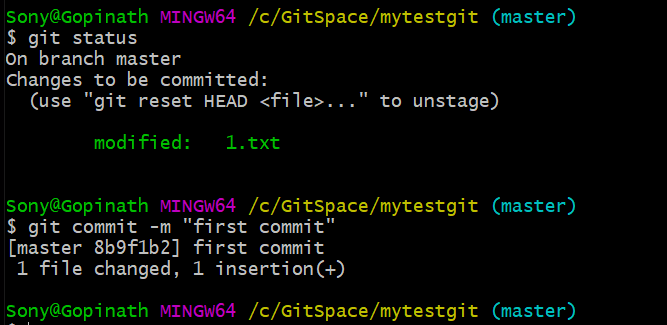
git status -> see th status first and then add to index

git add 1.txt



Now commit the changes

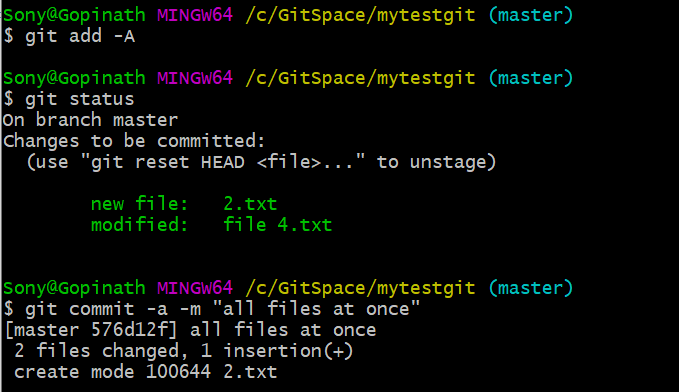
git commit -m “first commit”



To add and commit multiple files,

git add -A -> capital A represents adding all the files to index

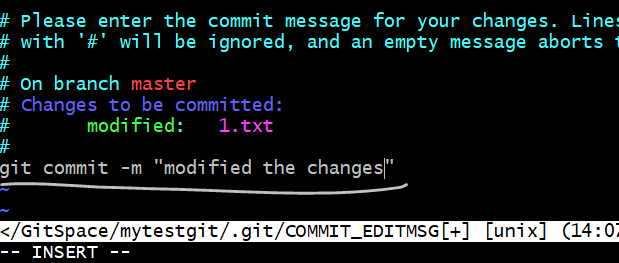
git commit -a -m “commit message” -> small ‘a’ commit all the files added to index and ready to commit.



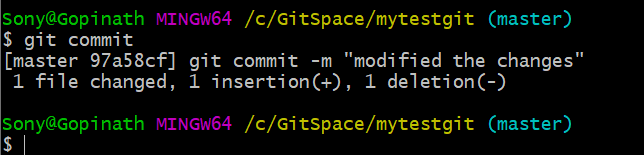
Cannot commit a file without message, even if you try to commit window prompts to enter message. Enter the message and :wq to exit the window prompt.



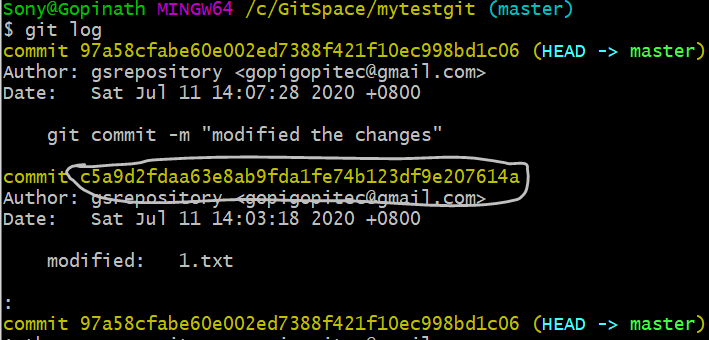
Insert the command with message



Press esc and enter :wq



git log – gives the detailed log of the commits with a 40 digit hexadecimal code.



1. GIT Branching

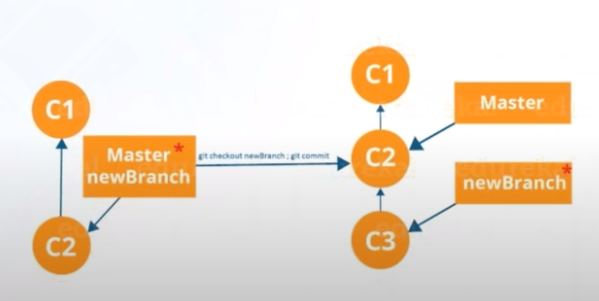
GIT is the only one have this feature which takes few seconds to create and merge branches

Allows to move the code back and forth between the different states/versions of the branch.

Lets you create a new branch to test new feature without affecting the main branch. Once done then easy to merge to main branch

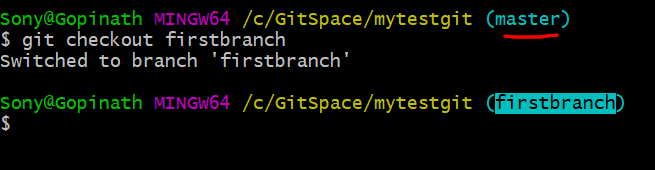
Branches are reference to a specific commit.

1. Local Branch
2. Remote-tracking branch

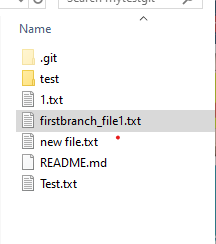


git branch firstbranch – creates a new branch

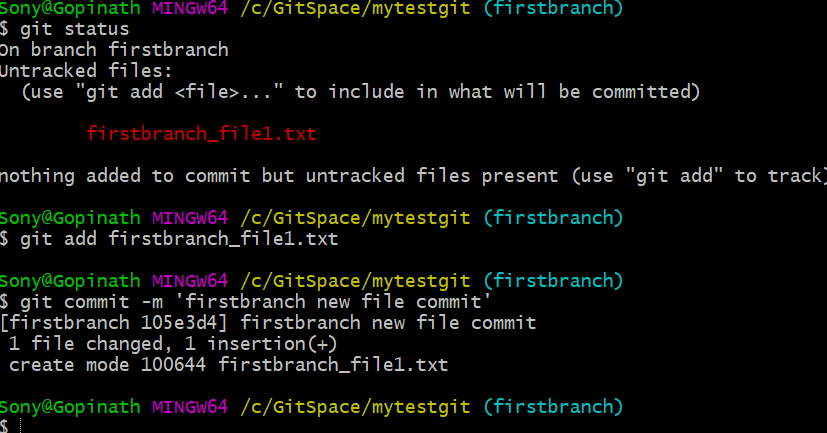
git checkout firstbranch – checkout and go to this branch



Now creating any file will be in new branch ‘*firstbranch’*

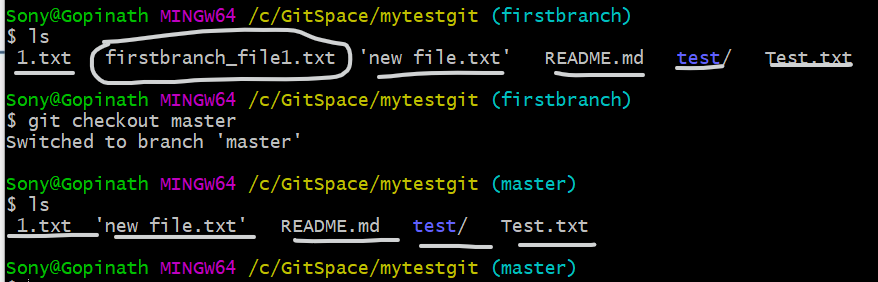


Create new file firstbranch\_file1.txt. Now add and commit to new branch



Now list all the files in the branch and see

ls the files and compare with *master* branch, in current branch *firstbranch* is having our new file, but this file is not present in *master* branch.

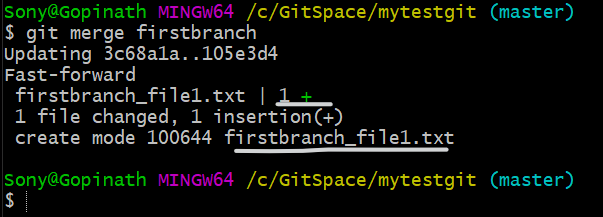


1. Merging

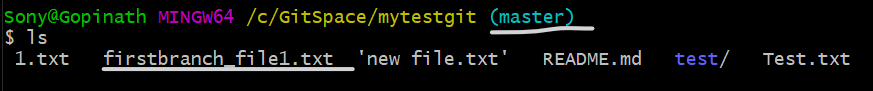
Now going to merge both the file

Make sure you are in *master* branch or the destination branch

git merge firstbranch



New file is listed in master branch

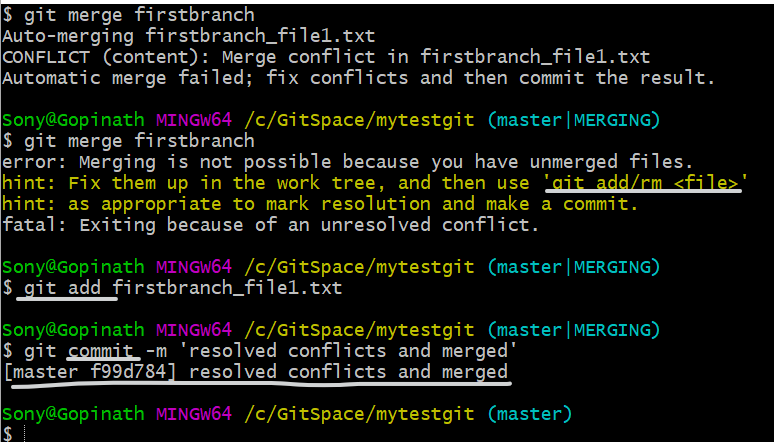


Any changes in the same file or other files, it won’t affect the *master* branch until its merged. Let’s change something *firstbranch* and merge again to *master* branch

see the highlighted diff contents in the branches

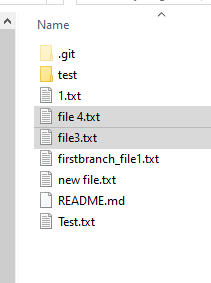
Changing the same file will resulting in conflict. We have to

1. Resolve the conflicts
2. Add again to branch tree
3. Commit

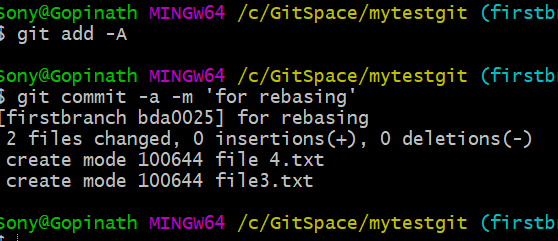


1. Rebasing
2. To merge the new branch to the tip of Master branch.
3. To get a much cleaner browsing history.
4. Reduce the number of branches, yes of course because having so many branches together might be confused if you don't merge.
5. To make the linear work flow, its good to do rebase and work.

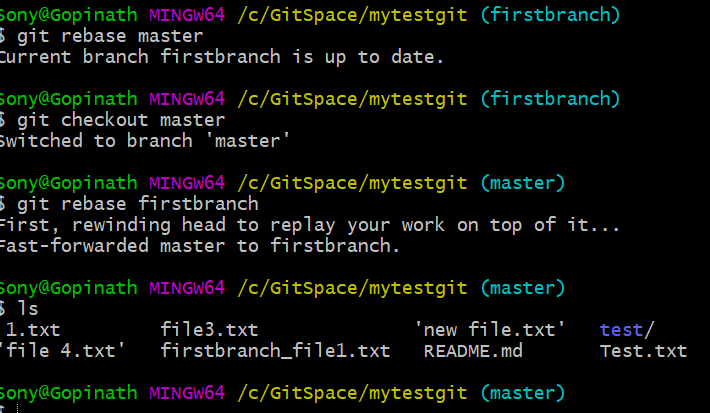
Let’s checkout *firstbranch* and create some new files,



Add and Commit now



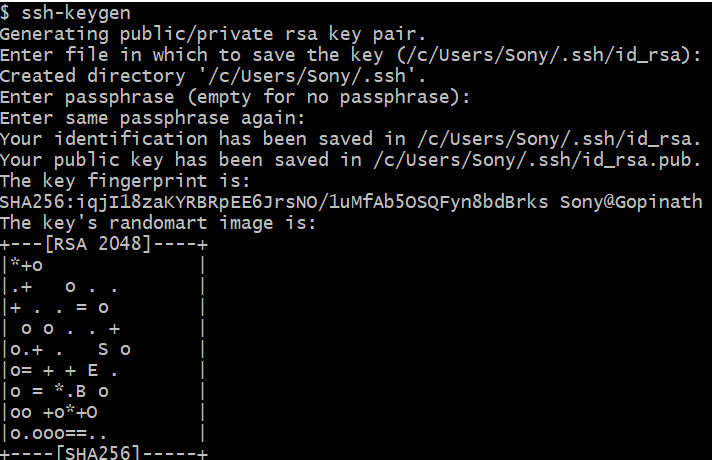
Now go to master and list files, and the new files in the *firstbranch* will not list in master branch here.



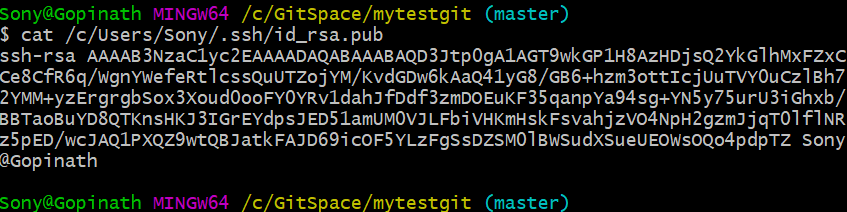
PUSH to Central Repository

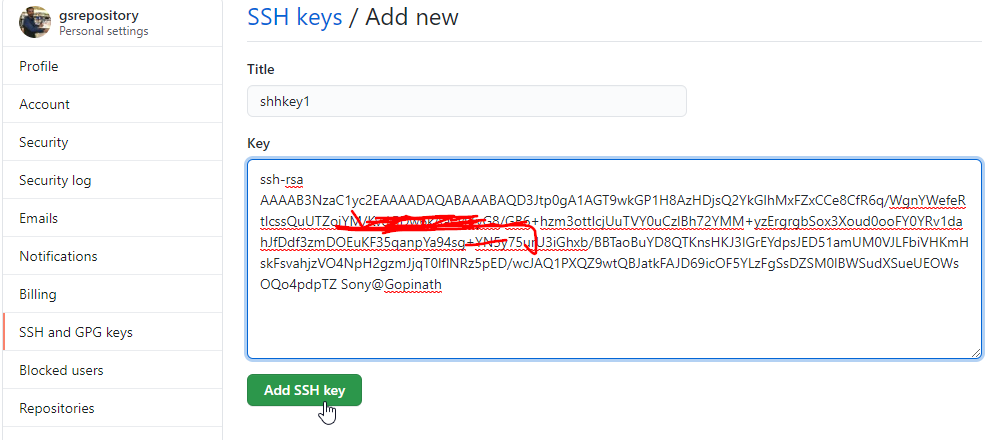
First get the ssh key and give access to the central repository

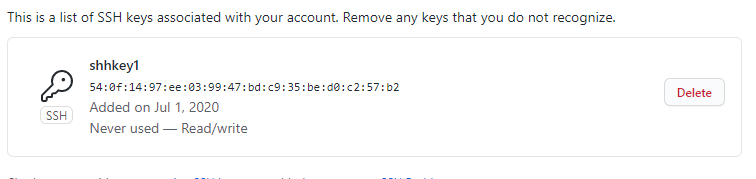
Generate ssh key as shown below, the key will be saved in a file.



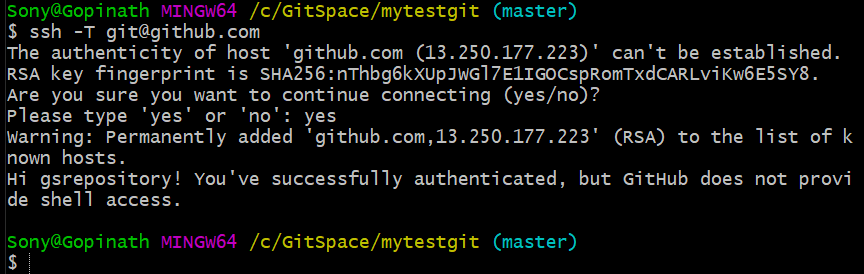
Copy the key and go to central repository

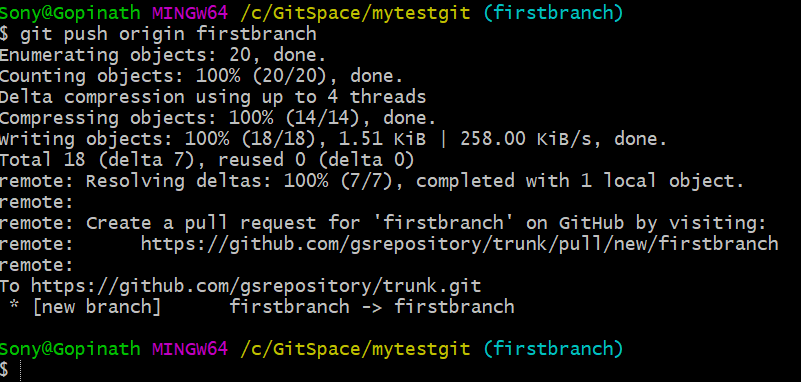


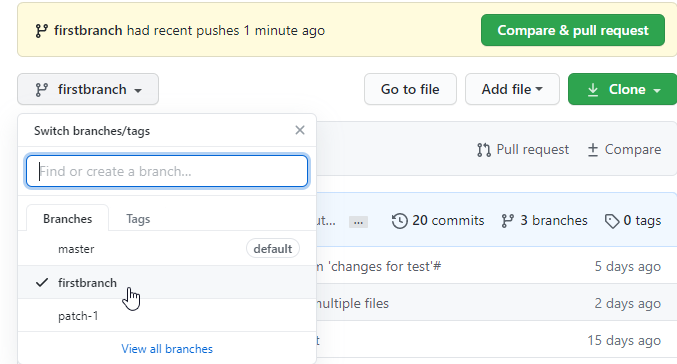




After adding it, authenticate it







Same way push the master branch as well.

