**GIT DOCUMENTATION**

Index

1. create git repository in your local desktop 2
2. Using git init command 2
   1. Create one file 3

2.2 git status command 3

* 1. git add 3
  2. git commit command 4

2.6 git log command 4

1. git revert 6

3.1 git reset --soft command 6

3.2 git reset –head command 7

1. create github account 7
2. local repository to github 8
3. clone the github 9

6.1 Copy url from github 9

6.2 Clone the repo into your local 9

6.3 How to create branch from master and checkout 10

1. git merge 10

7.1 git merge file 11

7.3 Check the merged branch changes are reflected on github 11

7.4 pull request 12

1. merge tool installation 13

8.1 The process of git merge tool 13

8.2 The result of merged file 15

8.3 commit modified files 15

1. git stash 16

9.1 Create a stash 16

9.2 git stash pop 16

9.3 delete the Stash 16

1. git Tag 17

10.1 create tags 17

10.2 tag commit and push 17

10.3 check tags in git hub 17

10.4 List all tags 18

10.5 Checkout tags 18

11 git pull and fetch 18

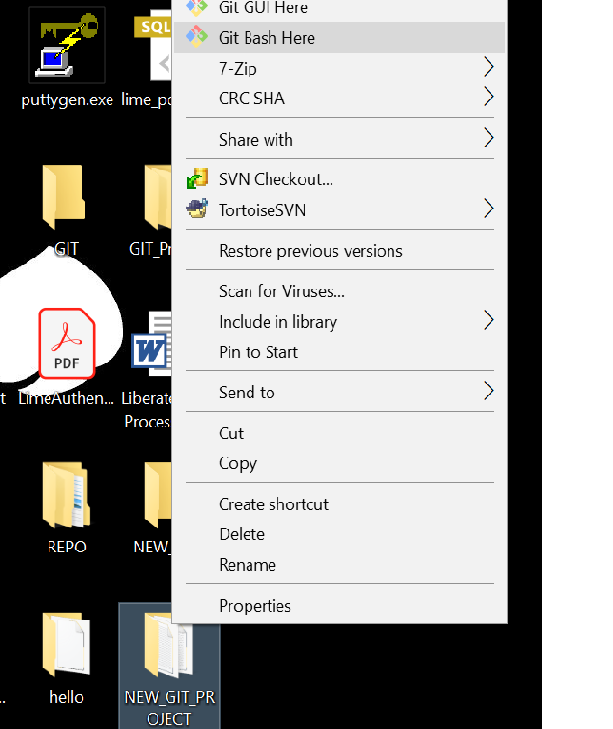
11.1 git pull 18

11.2 git fetch 18

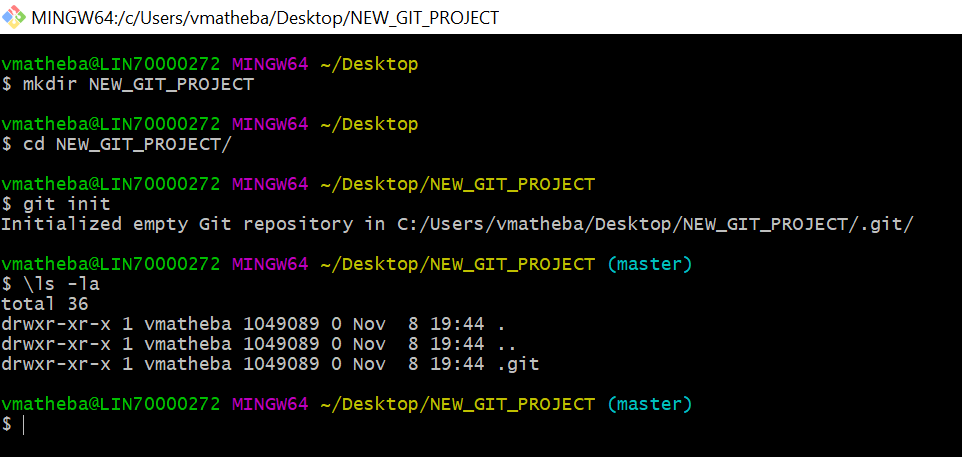
12 git commands 19

1. **How to create git repository in your local desktop:-**

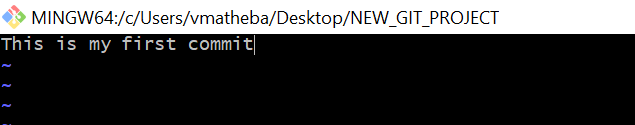
First download git package on your windows machine using: - <https://git-scm.com/downloads> and Create a folder (like:-NEW\_GIT\_PROJECT) on your desktop, open **git bash** and follow the below commands



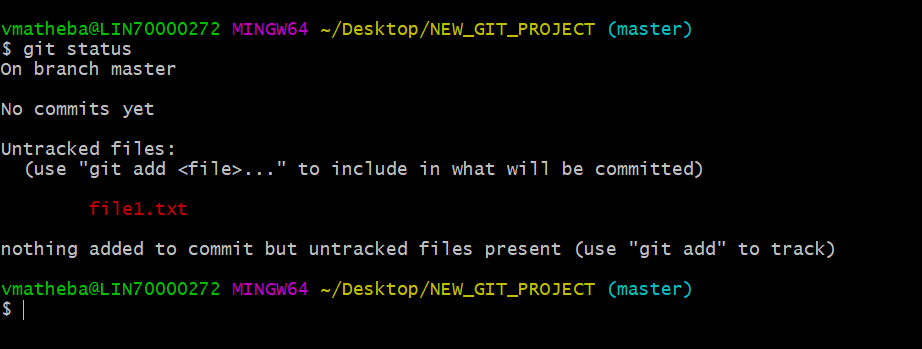
2 . Using **git init** command it will create .git repository into your **NEW\_GIT\_PROJECT** folder.



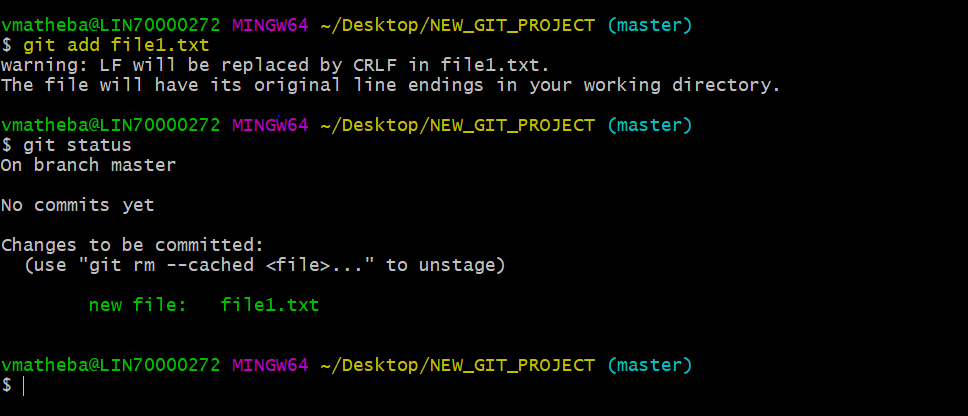
2.1 Create one file1.txt file using vi editor into git project and commit to your local git repository

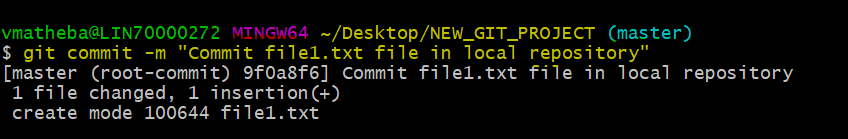


2.2 Before add check the status of current file using **git status** command, if it is in red colour, so changes is in work space area as shown in below screen shot.

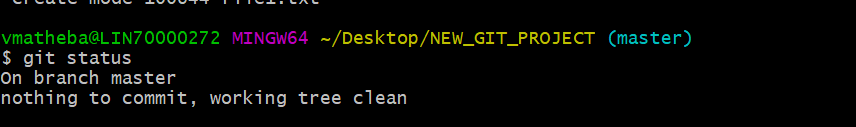


2.3 Add file1.txt file using **git add** command and check the file status(if you want to add more files using **git add . or –A or \*** commands), if file status showing in green colour, then it is in staging area so ready to commit into local repository.

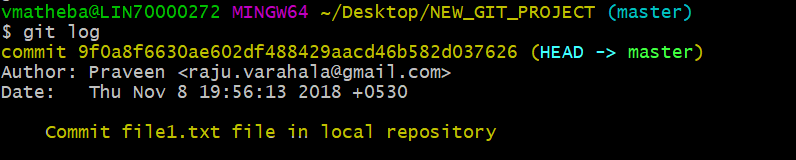


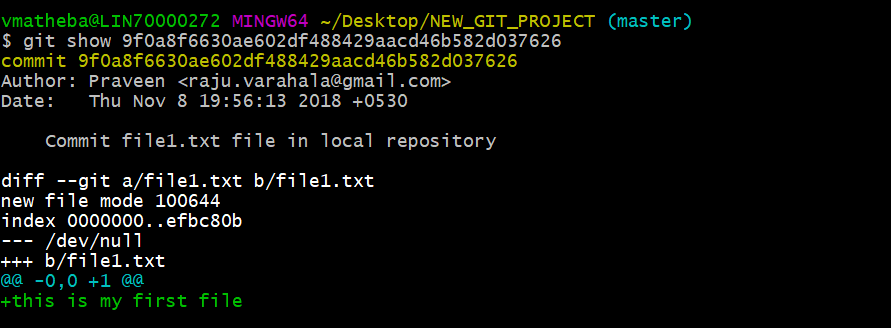
2.4 If you want to commit stagged files to local repository using **git commit** command, it will push the changes into your local git repository.

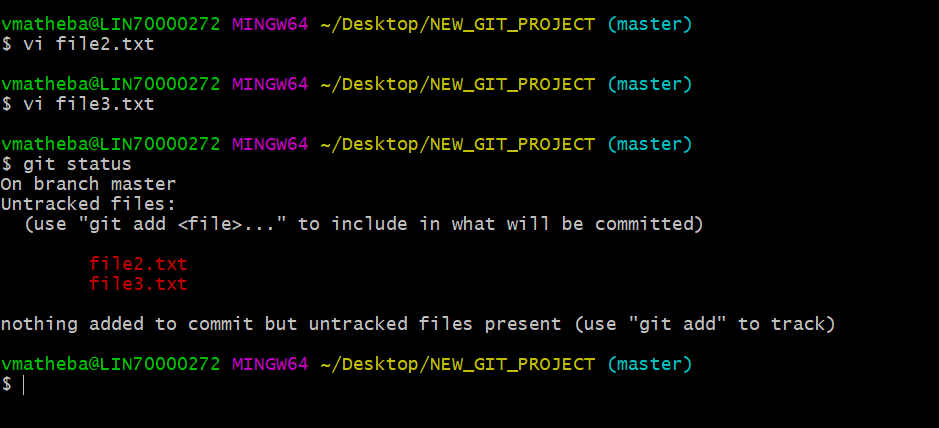
2.5 So as per below screen shot your working tree is in clean state (i.e:- All files in your workspace committed into local repository.

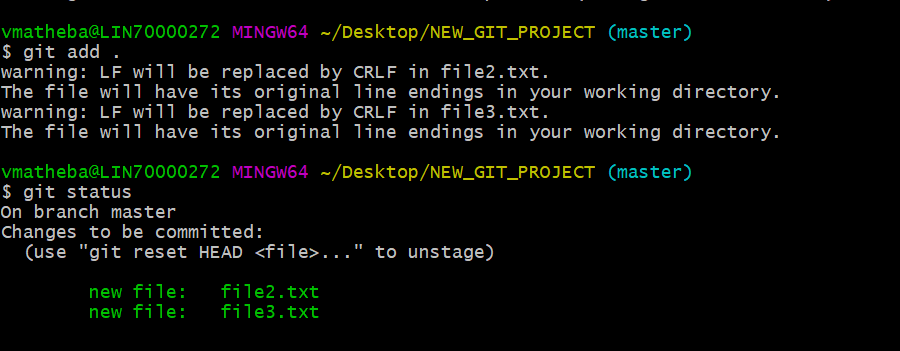


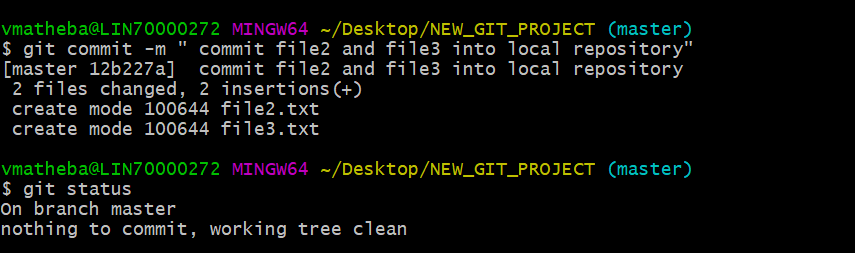
2.6 Once committed to local repository, then type **git log** command (because all files are saved in objects manner that is why git status is not working in this stage), it will show commit id. If you want to see which files committed under this commit id using **git show cid** command.

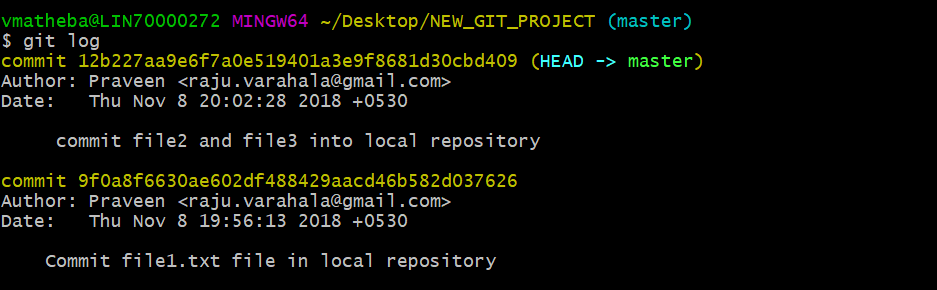




2.7 Now create another two files file2.txt and file3.txt using vi editor and follow the steps as per below screen shots(i.e: **git add and git commit**)

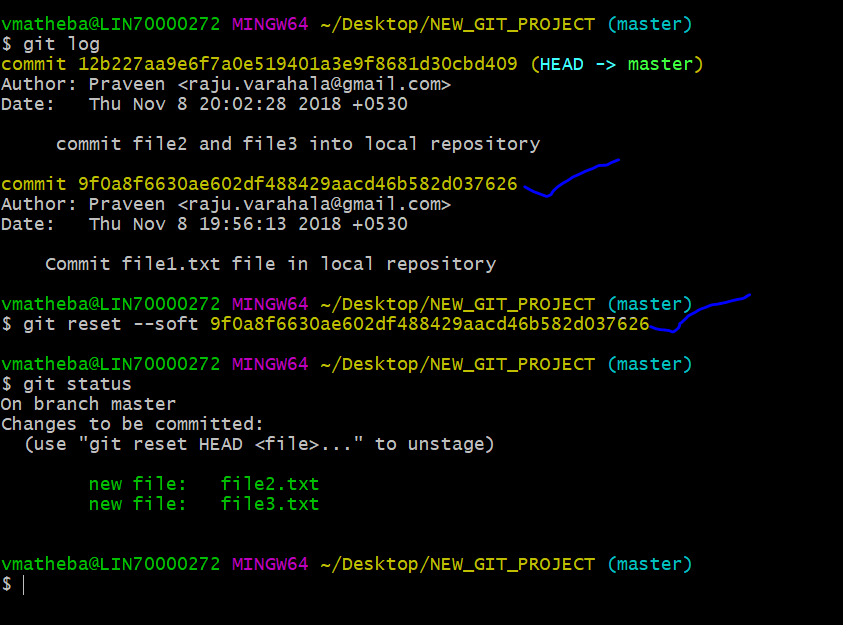


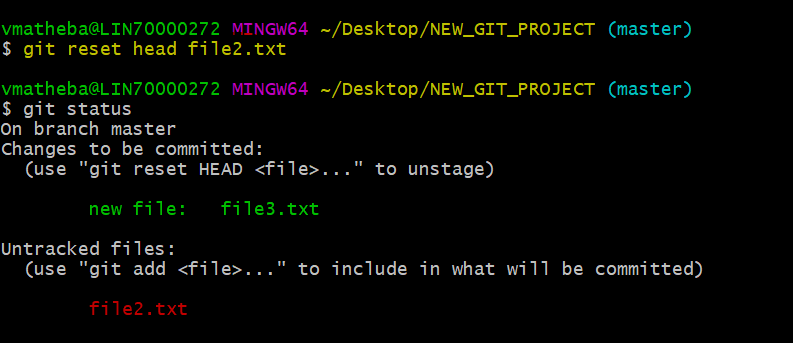




3. How to revert back the change from local repository to staging and staging to workspace(conversely)

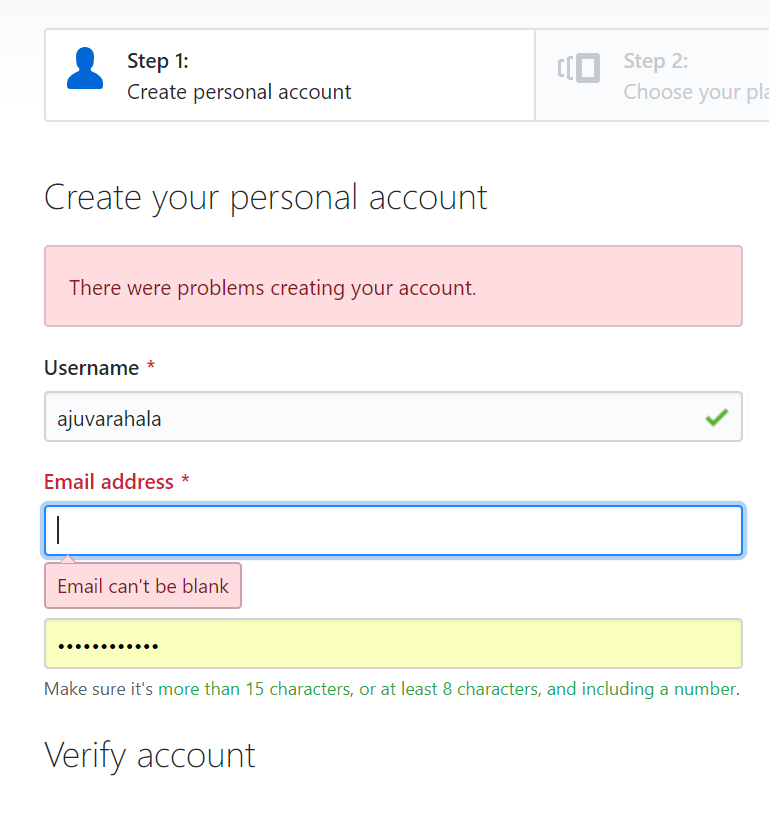
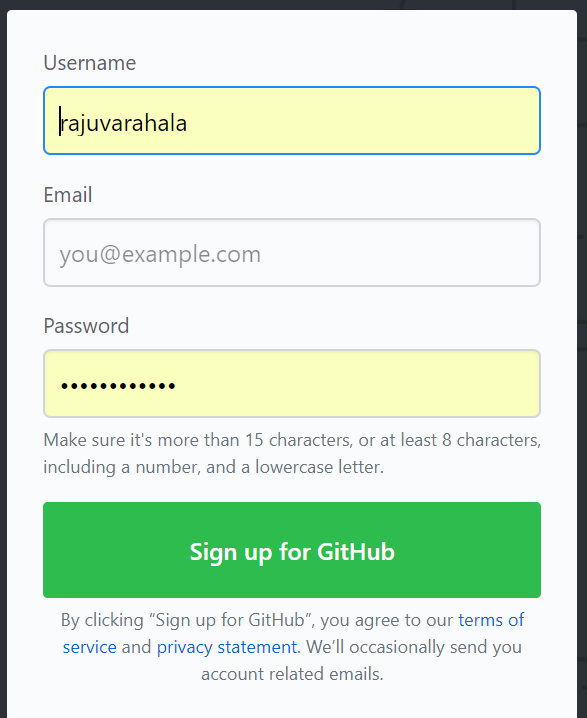
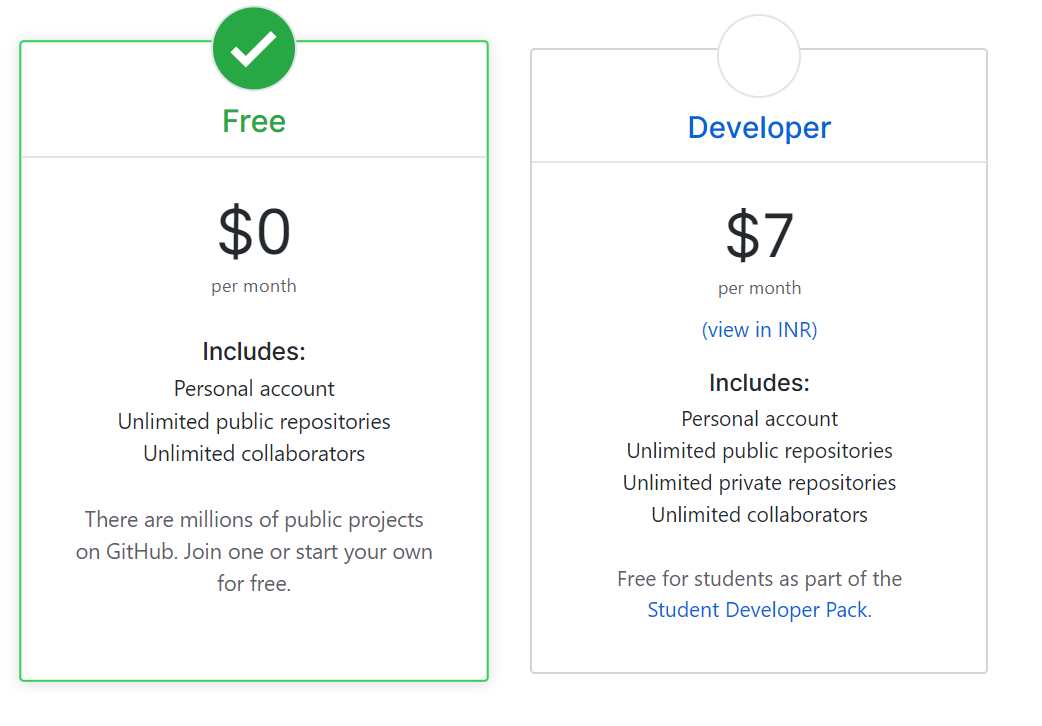
3.1 Using **git reset –soft cid**, it will revert back your local repository files into staging area, as shown in below screen shot.



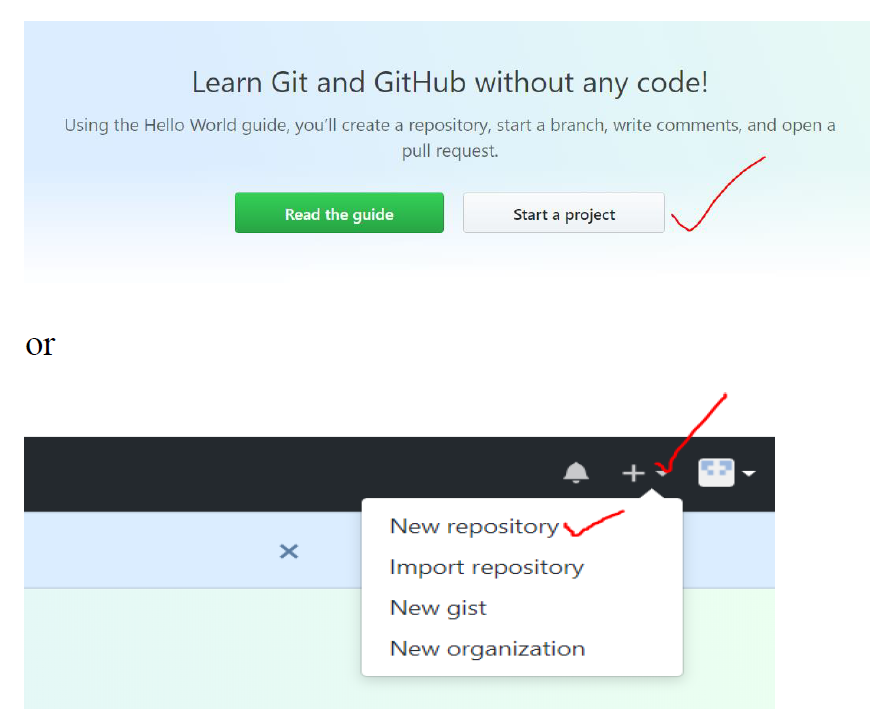
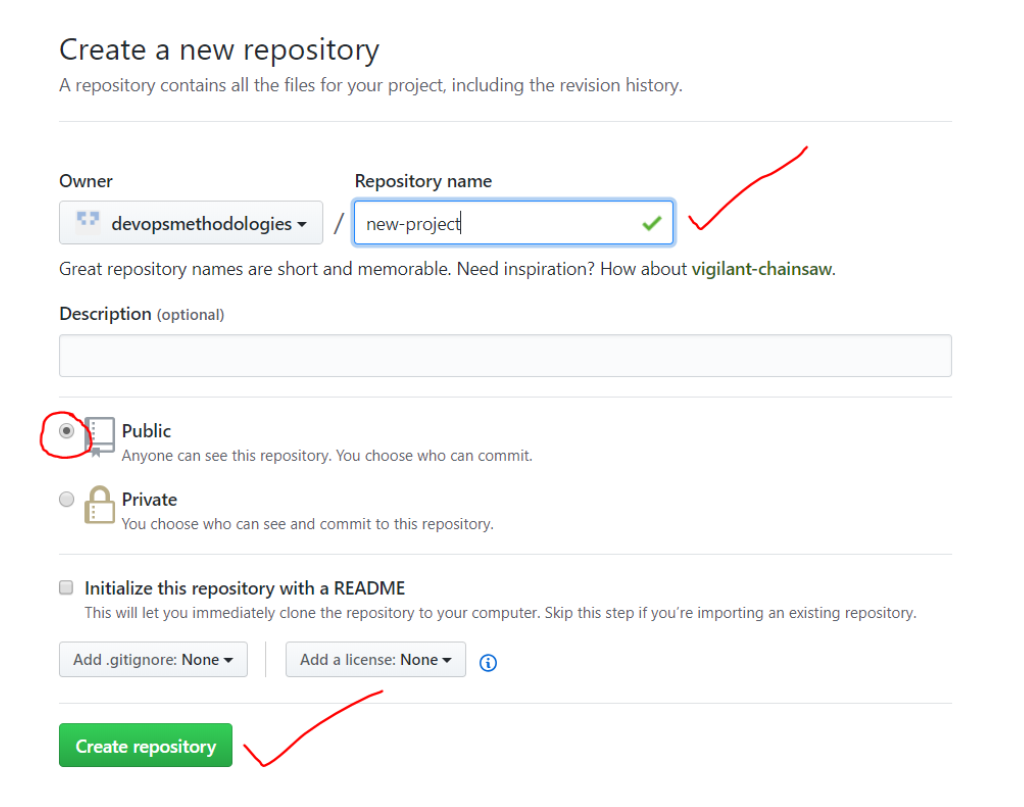
3.2 Using **git revert head file name**, it will revert back your staged files into workspace, as shown in below screen shot.

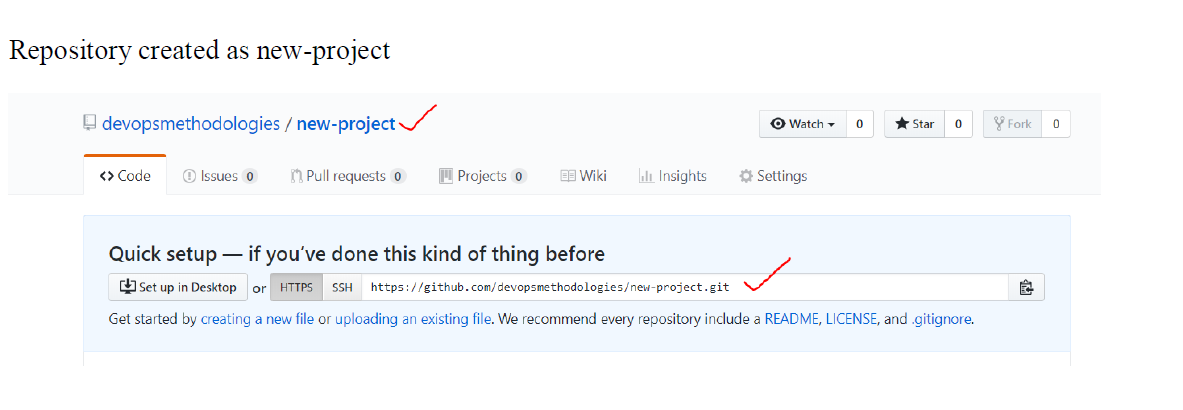
3.3 How to revert back local repository changes into workspace:- **git revert –mixed cid.**

4. **How to create git-hub account:-**



Create new repository on above signup git hub account



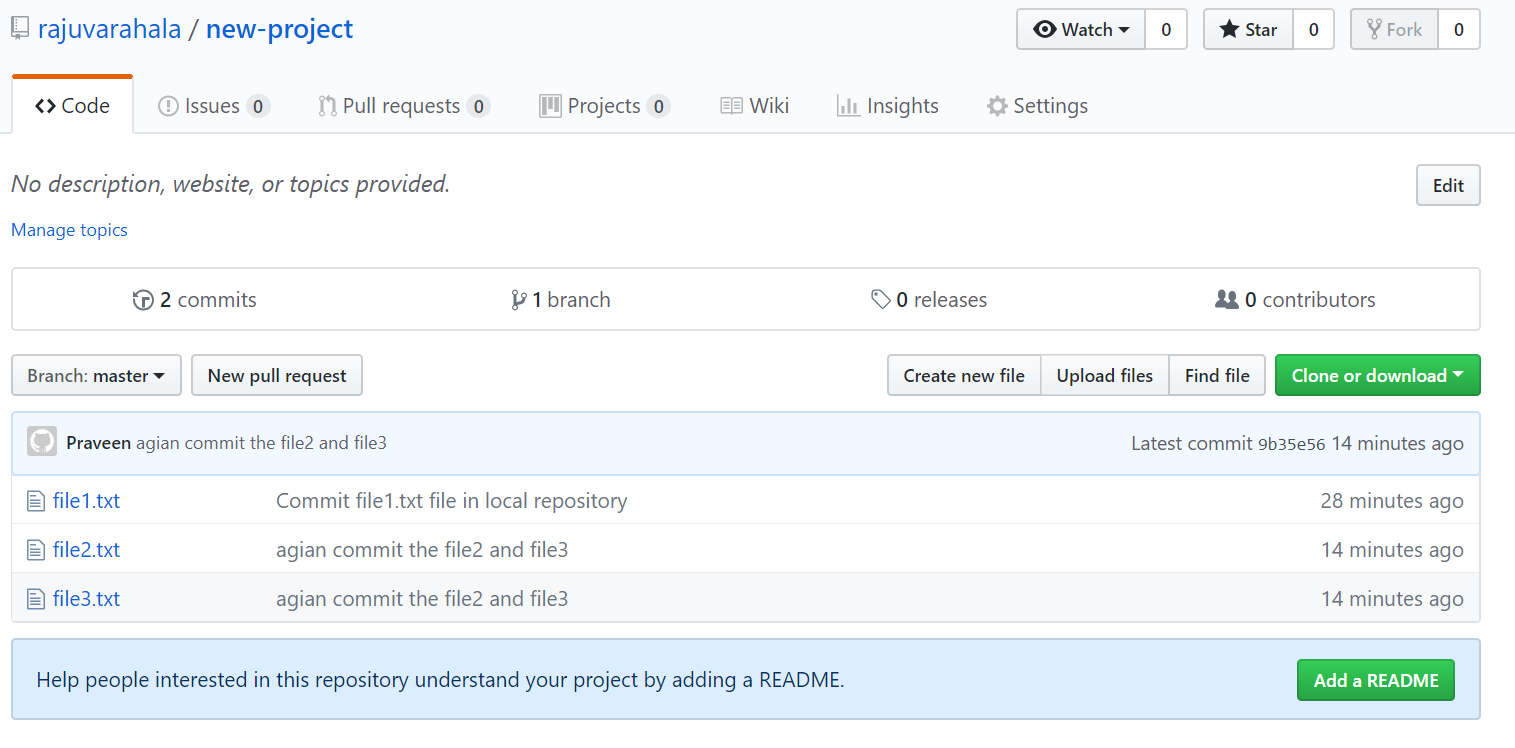
5. How to add your **local repository** to **github:**

5.1 Using **git remote add origin url** command as shown in below, it will automatically added created repository into github.

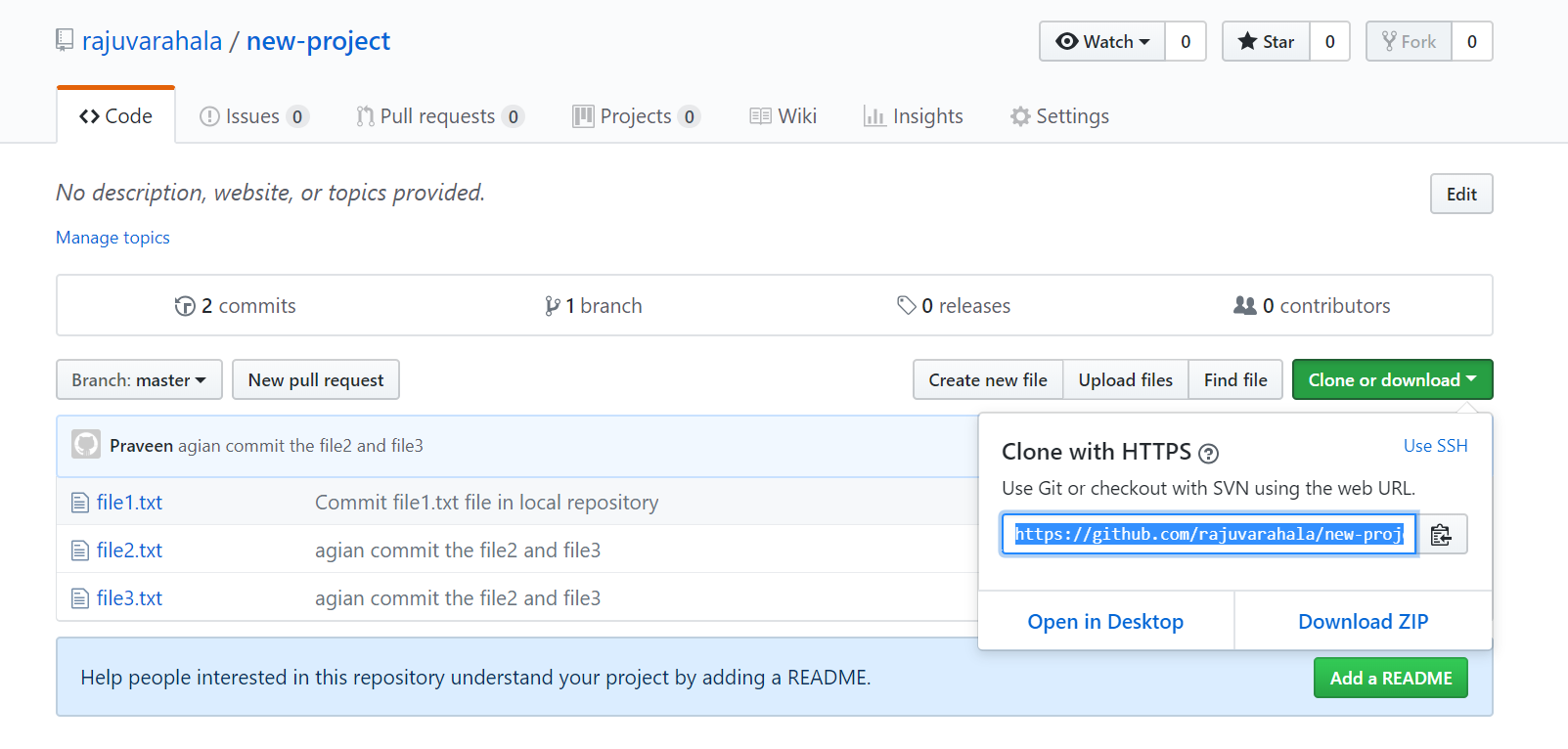
Note: - We have already created git local repository in (refer:-point 1) using **git init** command and same has been pushed into central git hub as shown in below screen shot. But first we need to create one empty repository on git hub and add your local repository with same name [In above point we have already created empty git repository on git hub (i.e:- new-project)].

Once added to origin and then need to push the same repository into origin master branch

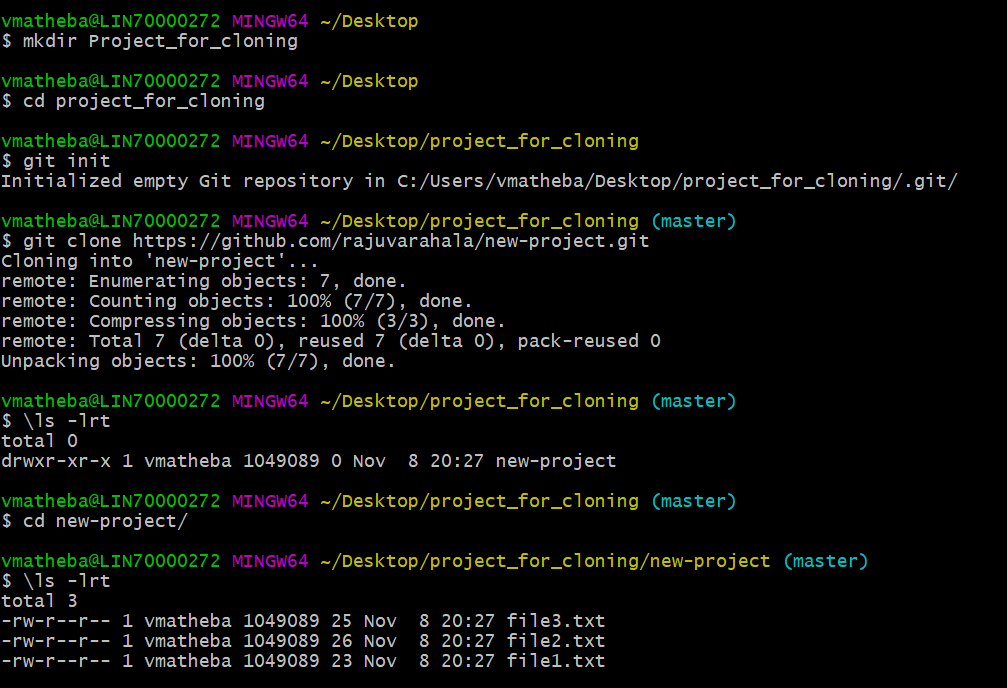


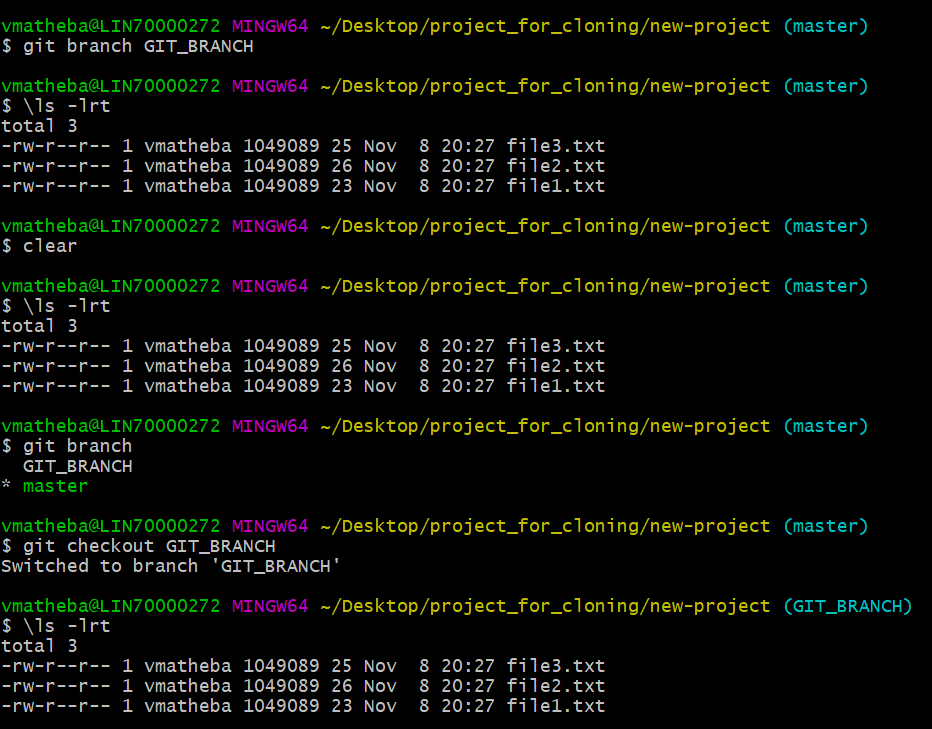
* 1. And go to **github** and check the above added repository are exist or not

6. How to clone the new created repository on Github and clone the same repo into your local

6.1 :-Copy url from github

* 1. :- Clone the repo into your local

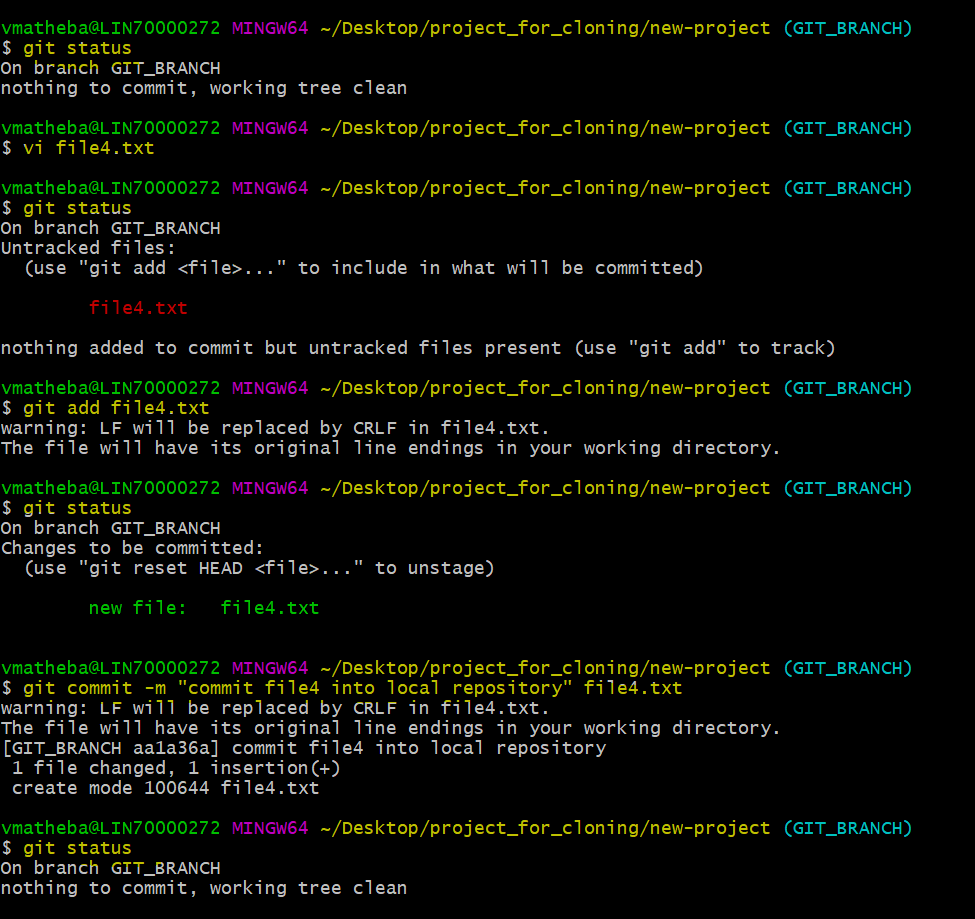


6.3 cthe same branch

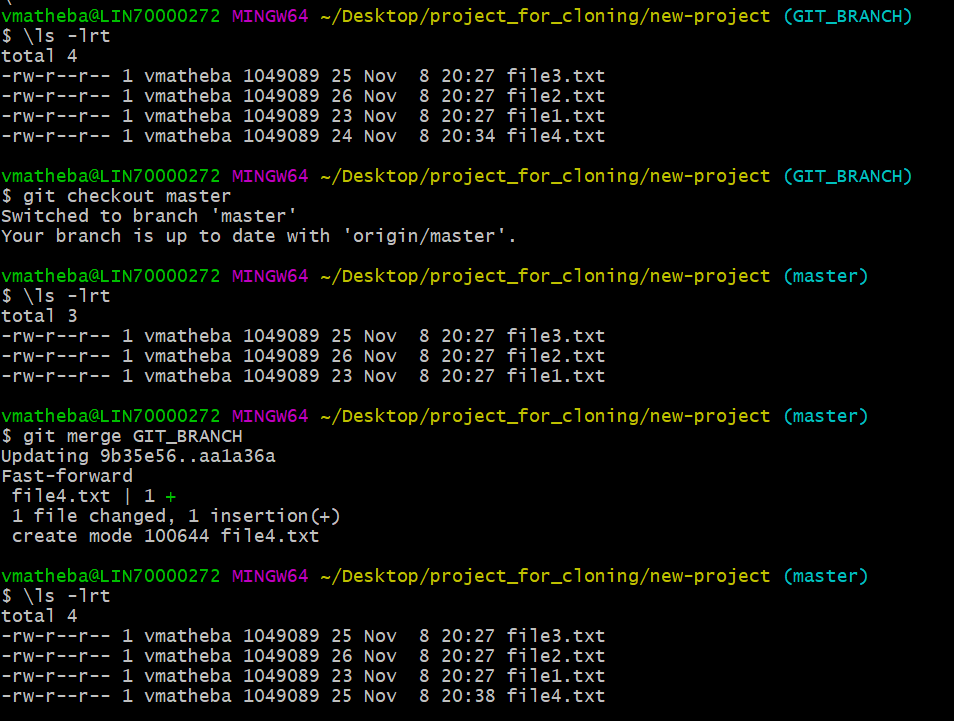
**7. Git Merge:**

When two developers tried to commit the same files with their changes then git

Conflicts arise. In order to resolve this one has to use **merge** tool and pull compare the changes and then push the modified file.

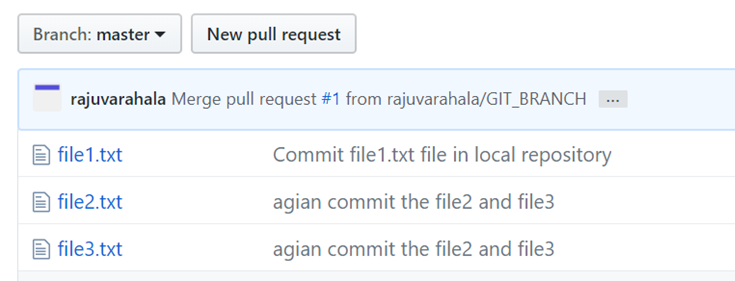
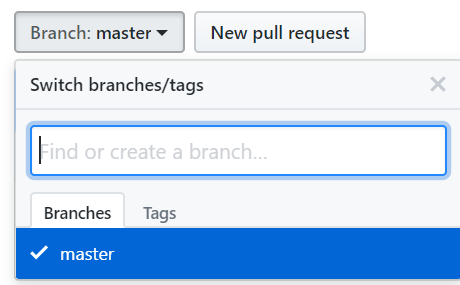
Checkout done, so now you are in GIT\_BRANCH then try to create one file and **merge** the same file into master branch.

7.1 GIT\_BRANCH has 4 files, but in master has only 3 files see the difference in below screenshot. But after perform the **git merge** file4.txt merged into master branch.

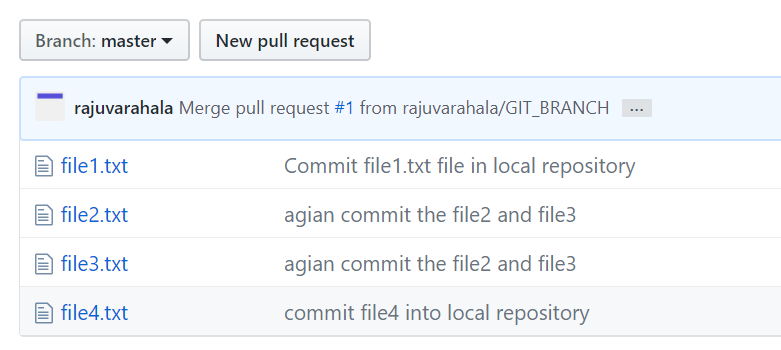


7.2 So all changes are merged in to your local repoand need to push the merged local master upto date changes into remote master using **git push origin master** command, then it will automatically pushed the merged changes into remote master branch.

7.3 Check the merged branch changes are reflected on **github** master branch? As per the below screen not reflected on git hub. We have only 3 files on master branch.



7.4 pull request



**8**. **Merge tool installation:-**

git config --global merge.tool p4merge

git config --global mergetool.p4merge.cmd "p4merge.exe \"$BASE\" \"$LOCAL\" \"$REMOTE\" \"$MERGED\""

git config --global diff.tool p4merge

git config --global difftool.p4merge.cmd "p4merge.exe \"$LOCAL\" \"$REMOTE\""

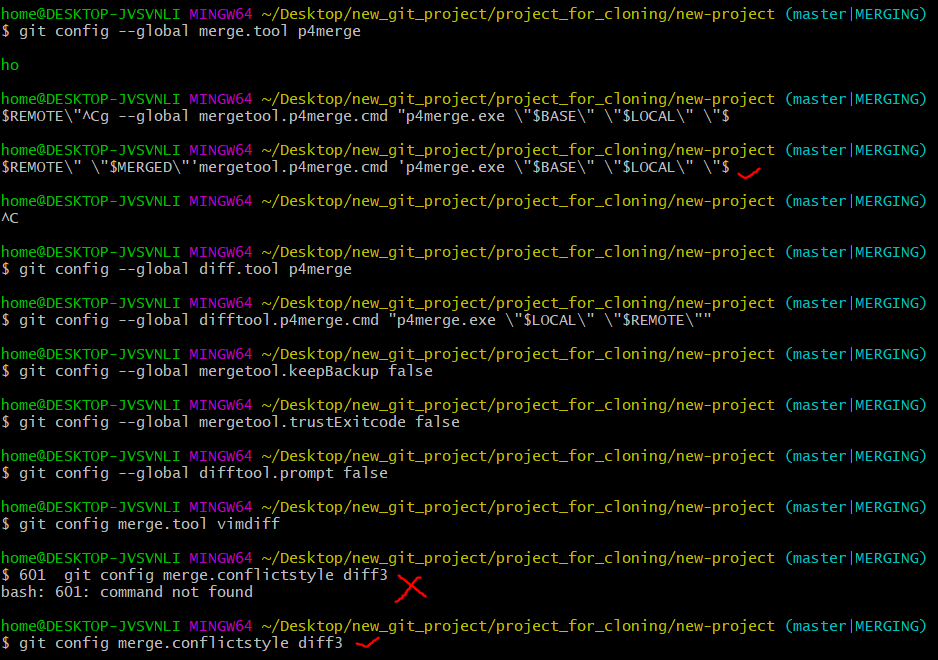
git config --global mergetool.keepBackup false

git config --global mergetool.trustExitcode false

git config --global difftool.prompt false

git config merge.tool vimdiff

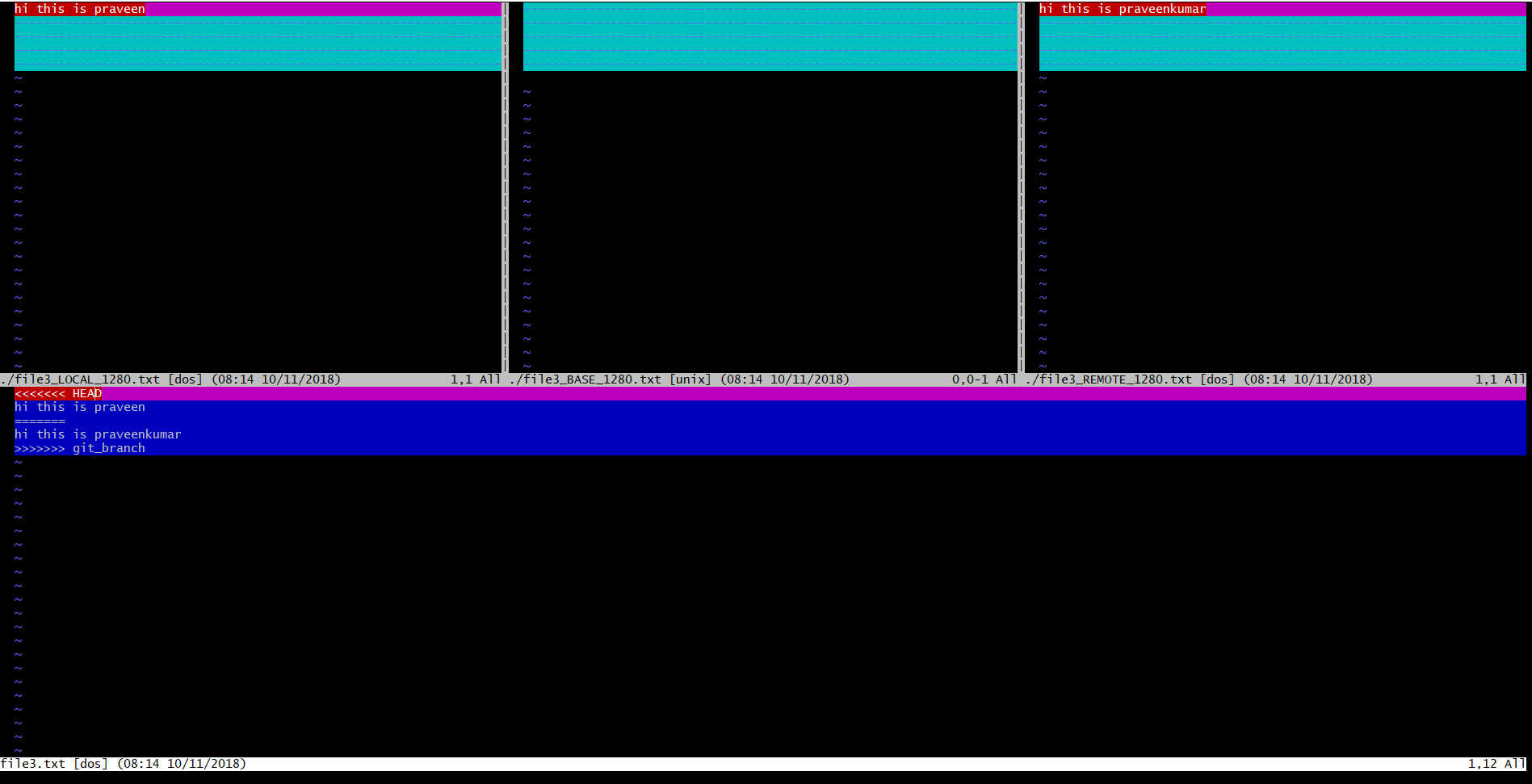
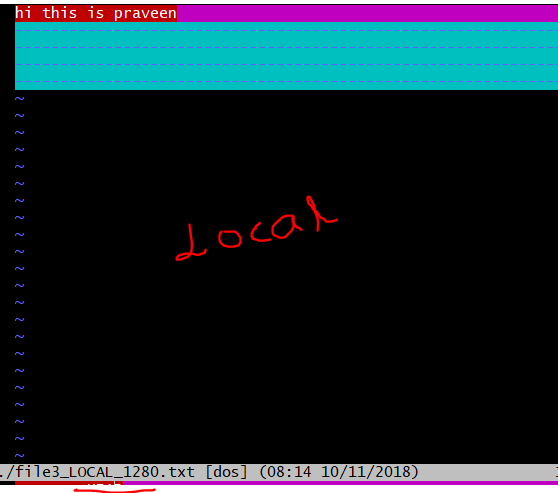
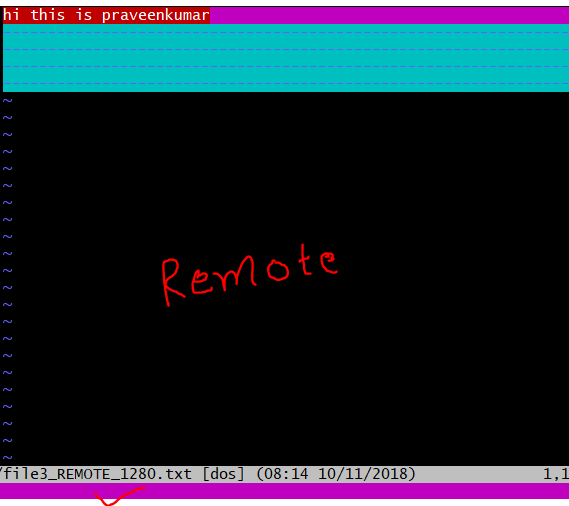
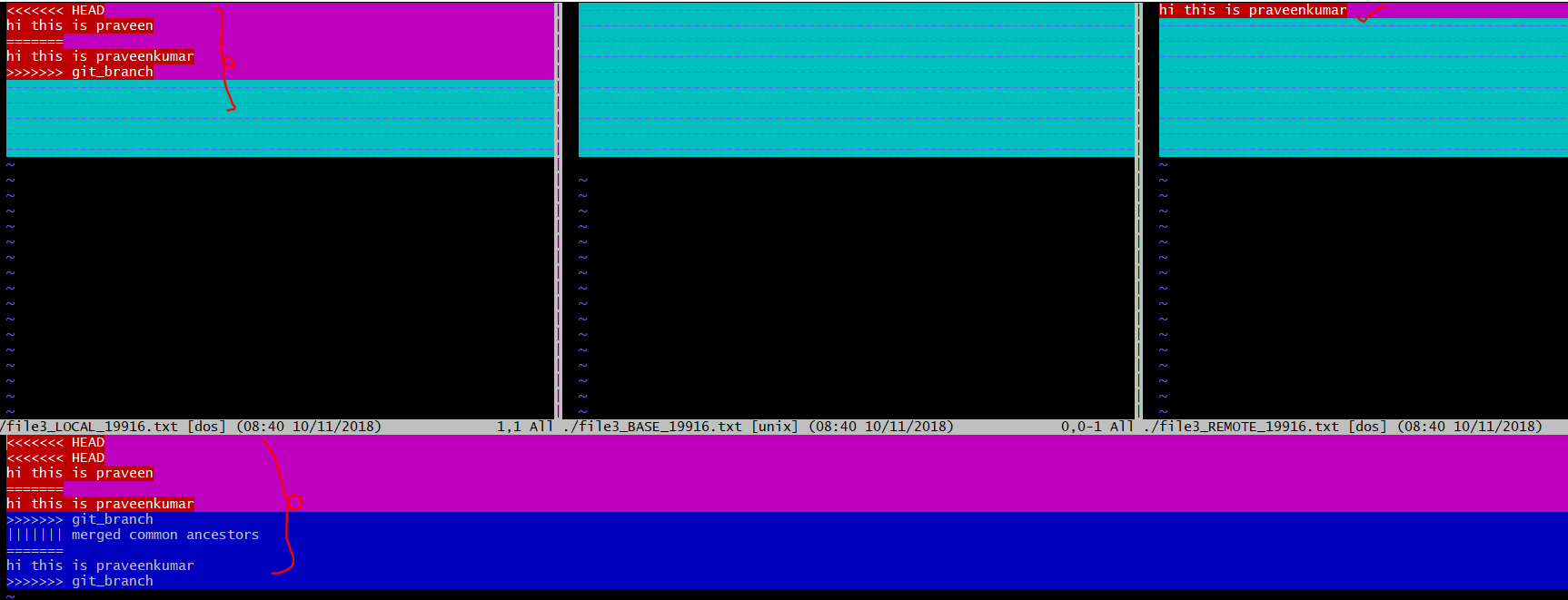
git config merge.conflictstyle diff3

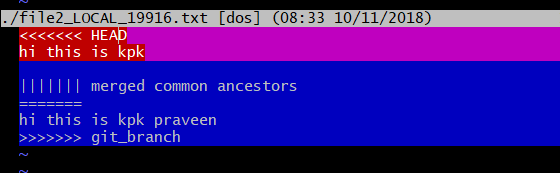


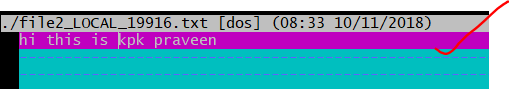
8.1 The below screenshots tells that the process of git merge tool. Openup merge window.

Its contain 4 text box

1. local
2. Base
3. Remote
4. Merge display

Select respective changes then save and quit   

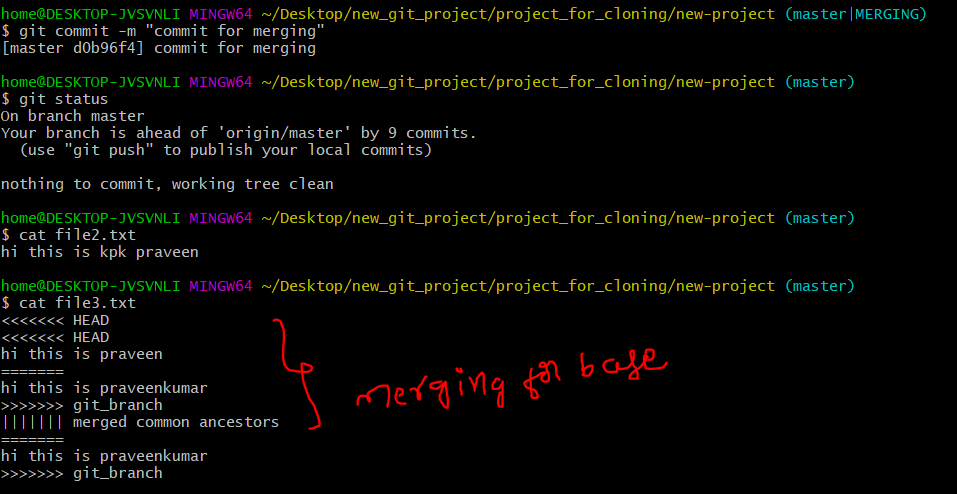




8.2 After successful exit from merge tool window, merged modified files displayed. The below screenshot is the result of file2.txt and file3.txt.

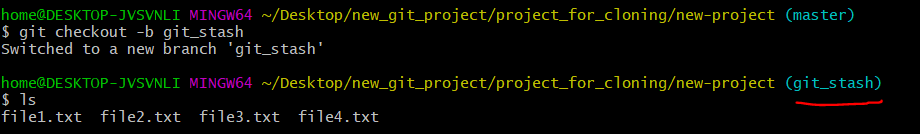


8.3 Then commit modified files.



**9. Git stash**

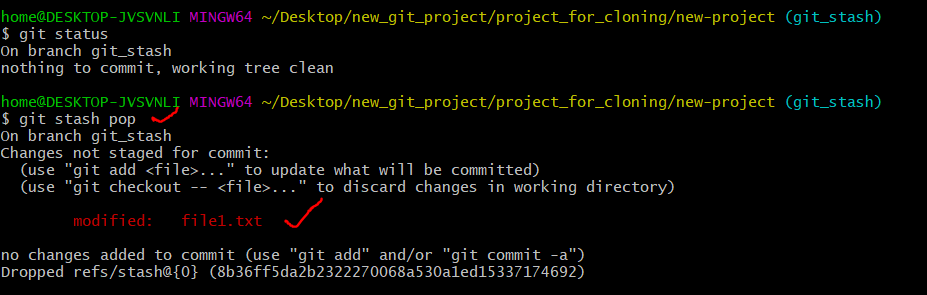
Stash stores the unstaged(git add) changes into a packages and stores in memory, whenever you want to apply the unchanged changes you can do this.



9.1 Create a stash with command using **git stash**



9.2.Check **git status**, it displays your **git\_stash** branch has been clean and there is no files on your local repo to commit( it means git stash creates separate virtual space for your changes and placed into stash repository).If you want to revert back the previous changes into git\_stash branch use below command **git stash pop.**



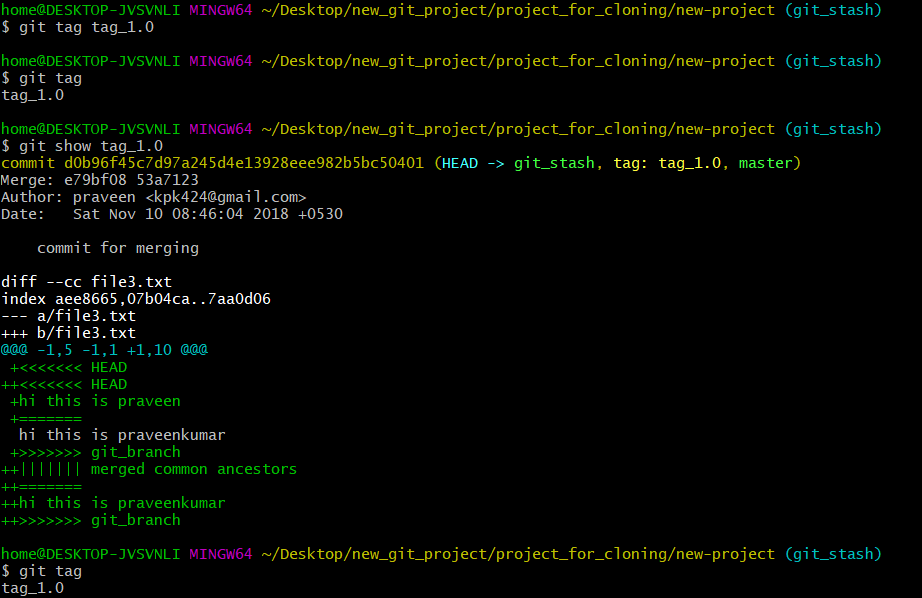
9.3 How to see the list of stash files using **git stash list** command, it will show the all stashed files with refs id’s and if you want to delete particular file using below **git stash drop refs id**.

How to delete the Stash: - **git stash drop stash@{0}** command.

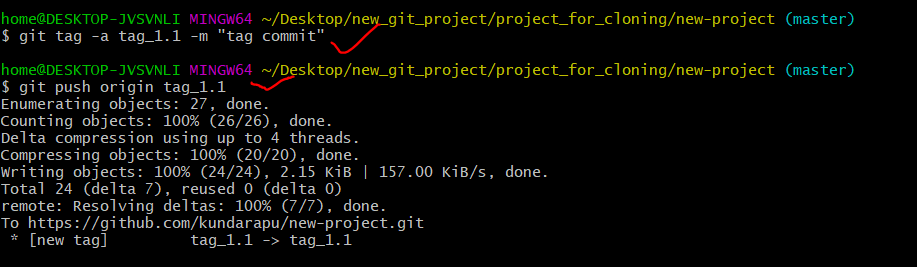
**10. Git tags**

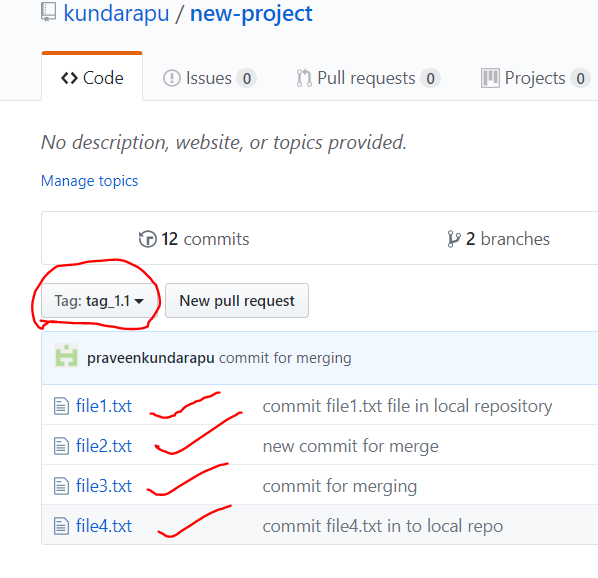
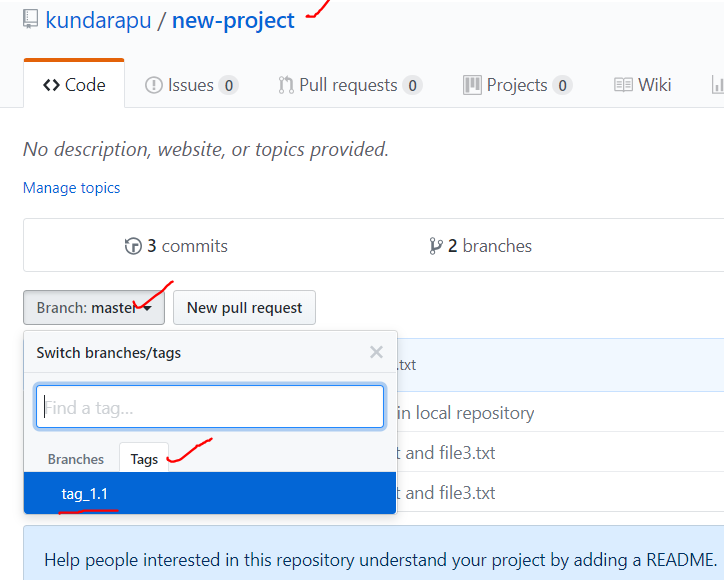
In git tags are read only, Snapshots of particular version number having a name and meant for read only.

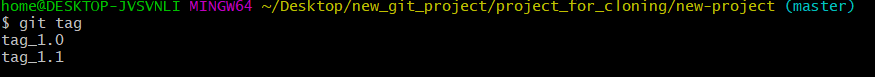
10.1 Create a lightweight tag then check the version with git show tag command and list all tag with git tag command.



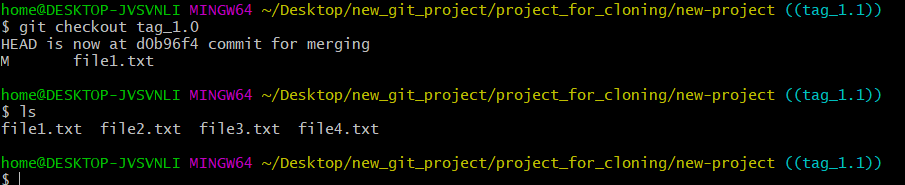
10.2 Create an annotated tag with some messages. And push to origin (github).



10.3 Then check tags in git hub to follow below screenshots10.4 List all tags



10.5 Checkout tags

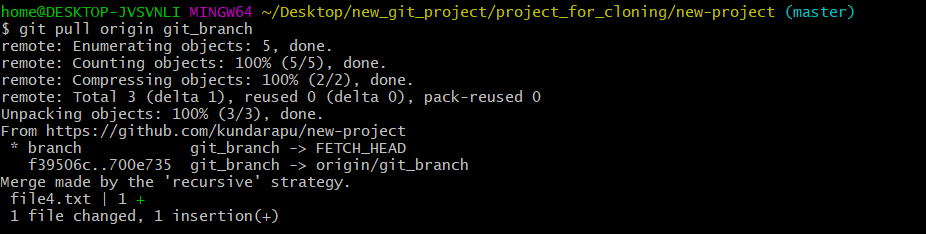


**11. Git pull/fetch**

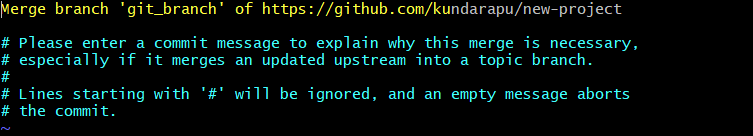
11.1 **Git pull:** Fetches files from remote repository and integrates with local repository

Note: git pull fetches and merges the remote repository to your local repository, whereas git fetch only stages the files into local repository index.

Enter git pull command as shown below.

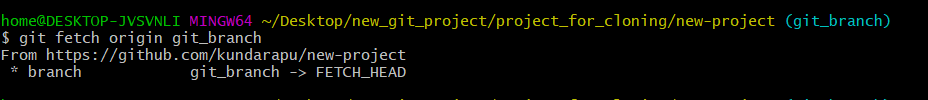


Below merge window contains details of commits that need to merge. Save and quit



**11.2 git fetch**: This stages the files of remote repository in the local repository.

Enter the below command and check the result then compare with git pull result.



**12. Git commands:**

* Git config –list(all env variable of git)
* Git config –global user.name Praveen(set the user name as praveen)
* Git config –global user.email Praveen@gmail.com(set the user email)
* Git config –global push.defaul simple
* Git clone
* Git add
* Git commit –m “message”
* Git commit –a –m “”(staging and commit)
* Git push(send the commit to remote repo)
* Git status(status of working tree)
* Git branch(on which branch you are in)
* Git pull(get the changes from remote repo)
* Git mergetool(this will open your merge tool)

🡪 git config --global merge.tool p4merge(mergetool setup first command)

🡪 REMOTE\" \"$MERGED\"'mergetool.p4merge.cmd 'p4merge.exe \"$BASE\" \"$LOCAL\" \"$( mergetool setup second command)

* Git rm(delete the file in a working tree and to reflect the same in server remote repo you need to commit and push)
* Git log (history of your repo)
* Git log file name
* Git log –author=name
* Git shortlog
* Git log –online
* Git checkout (change set)
* Git diff (to check the difference between staging area and git)
* Git show (show changes of patcset)
* Git clean –n
* Git clean –f
* Git blame file1(information about each line modification done by author)
* Git fetch/pull (track remote branching into local working tree)
* Git pull(performs a git fetch and git merge)
* Git merge(**git** branches and integrate them into a single branch.)
* Git reset (used to remove commits from the current branch.)
* Git branch(list all branches)
* Git stash(Stash stores the unstaged(git add) changes)
* Git tag(Snapshots of particular version number)