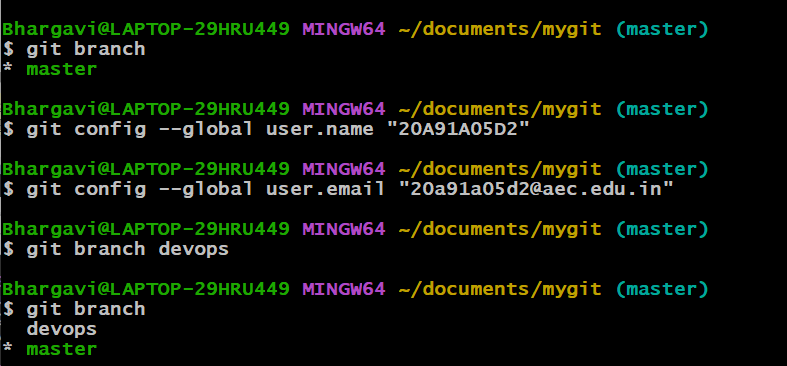
**Q1. Describe the usage of the git stash command by using an example and also state the process by giving the screenshot of all the commands written in git bash.**

**Git stash:** The git stash command takes your uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from your working copy.

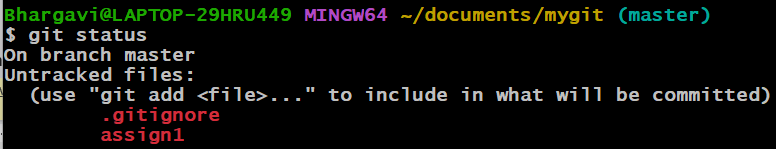


Create a file. For example here we created assign1.



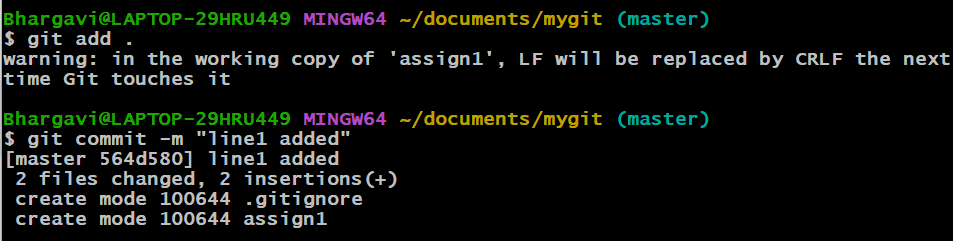


\*Use **git status** command to observe in which stage the file is in.



\*The file is in untracked state.

\*Now we are adding the untracked file to the staging area using **git add command** and then commit the changes using **git commit command.**

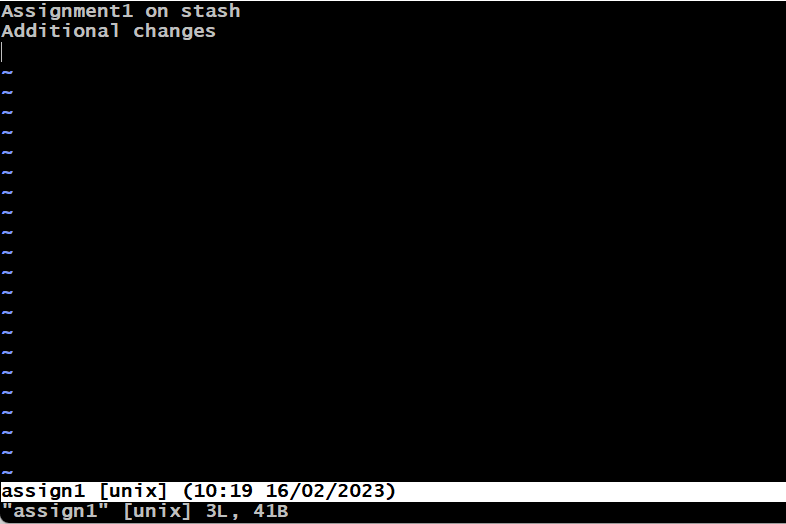
****

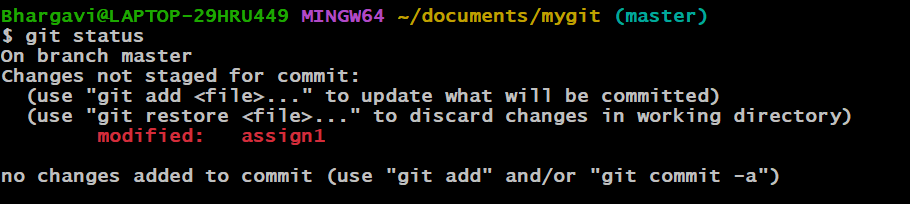
\*Let us consider the branch we are working be “br1” if there is a need to switch to another branch say “br2” then if there are any committed changes in br1 then these changes will be transferred/reflected to br2 which will disturb the workflow of br2.

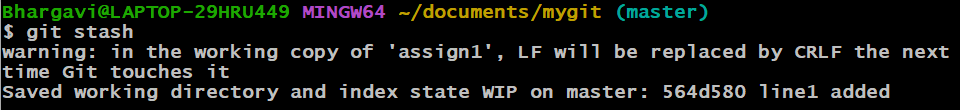
Inorder to avoid this we either need to remove the uncommitted changes. But if the changes are needed by us we need to place them in a separate place .For placing these changes we make use of “**Stash**”.

\*git stash (or) git stash save: Used to stash the uncommitted changes(staged or unstaged) into a stash stack. It undoes to the latest commit and place the additional changes(uncommitted) into a stack instead of deleting.

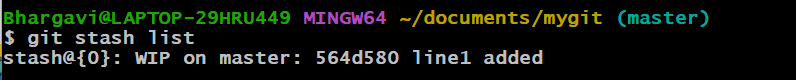




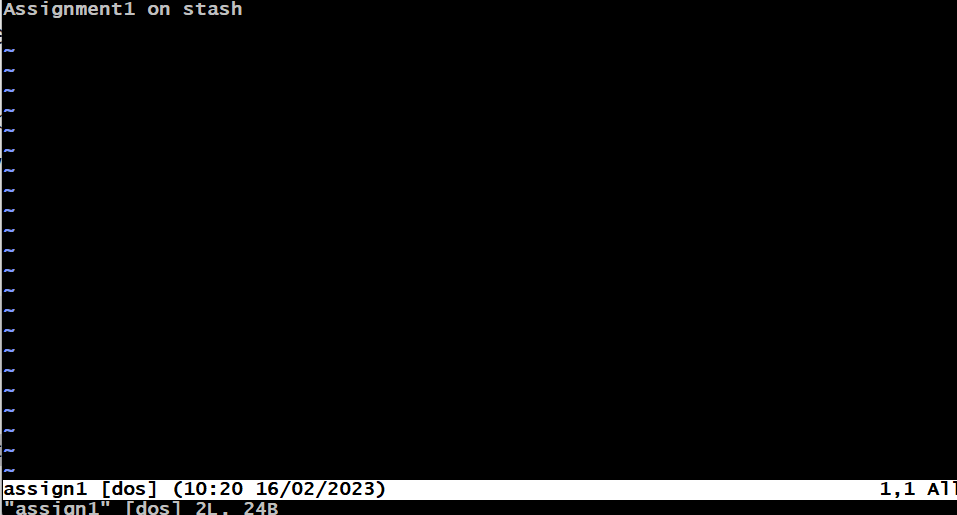




\***git stash list**: Used to list out all the stashes stored in the stash stack.







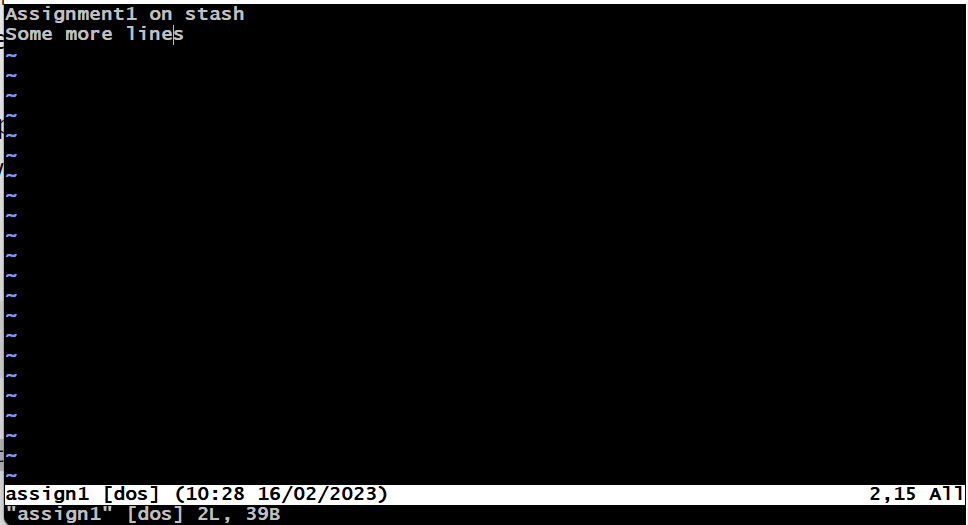
\*As we added the uncommitted changes to stash the line “Additional changes” got removed.

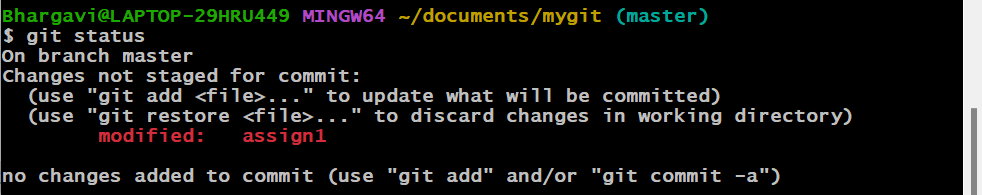
\*As the stack follows the LIFO(Last In First Out) the latest stashes will be at the top of the stack i.e, at 0th index.

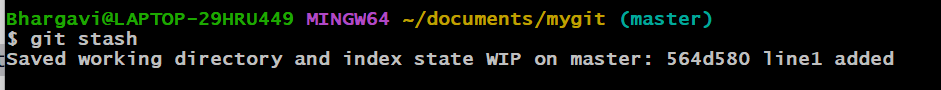
\*WIP stands for Work In Progress.

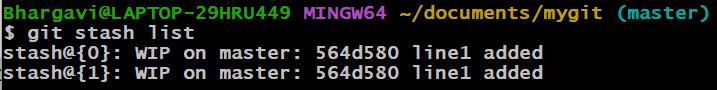
\*In the above example we get the latest commit id along with the commit message.





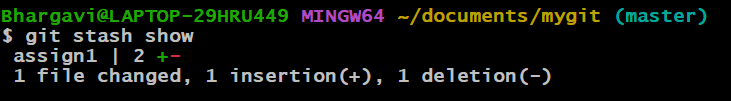






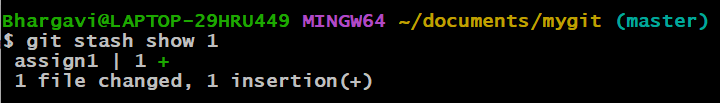
\***git stash show:** By default git stash show shows the changes recorded in the **latest** stash (stash@{0}) in the --stat format.

The --stat format shows only how many lines you've added and deleted on each of the modified files.

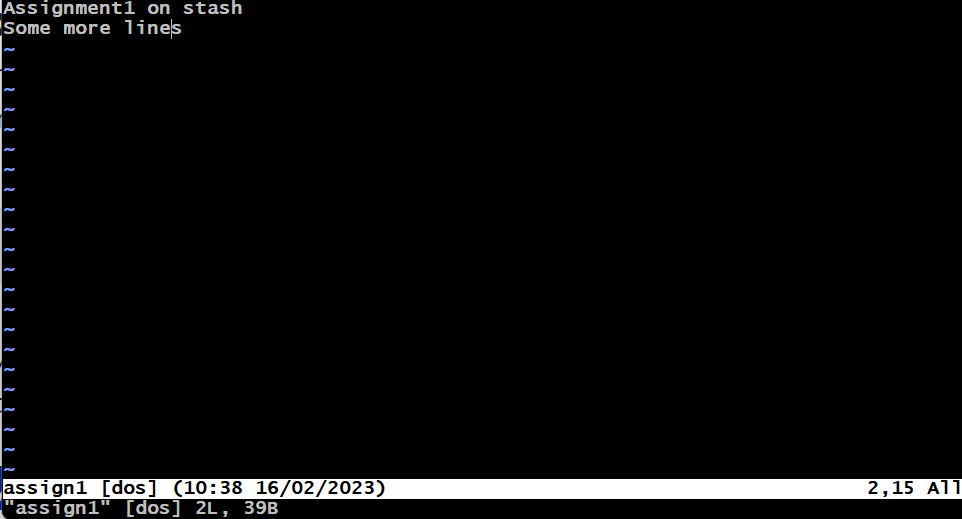
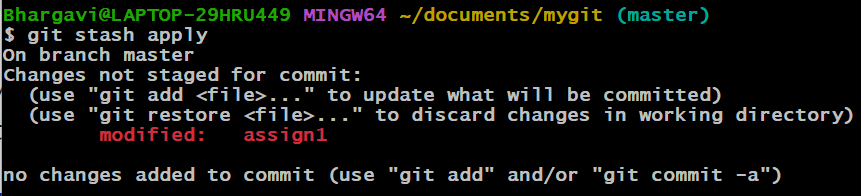


\*git stash show can also be used done along with the index number to show the changes at that particular index.

Syntax: git stash show –index

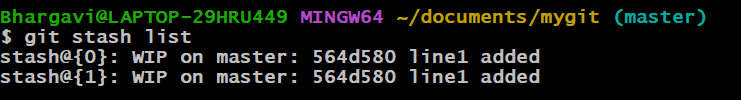


\***git stash apply:** Used to apply the latest stash without removing the stash from the stash stack.



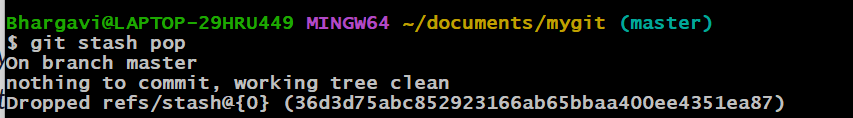
\*The changes from the latest stash i.e, the line “Some more lines” got added to the file.

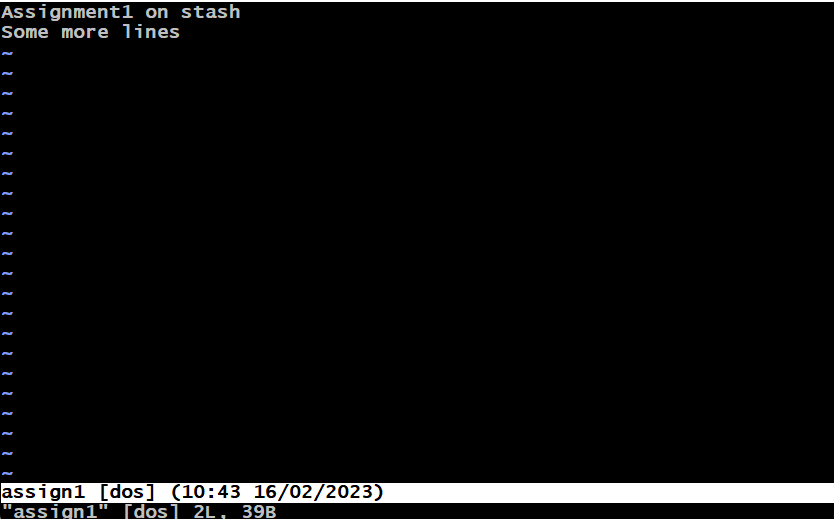
\*The stash was also not removed from the stack.



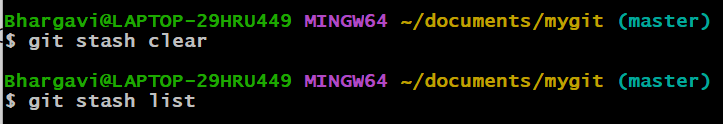
\*We can also use it along with index number by giving 🡪git apply index

\***git stash pop:** Used to apply the recorded changes of your latest stash on the current working branch as well as remove that stash from the stash stack.





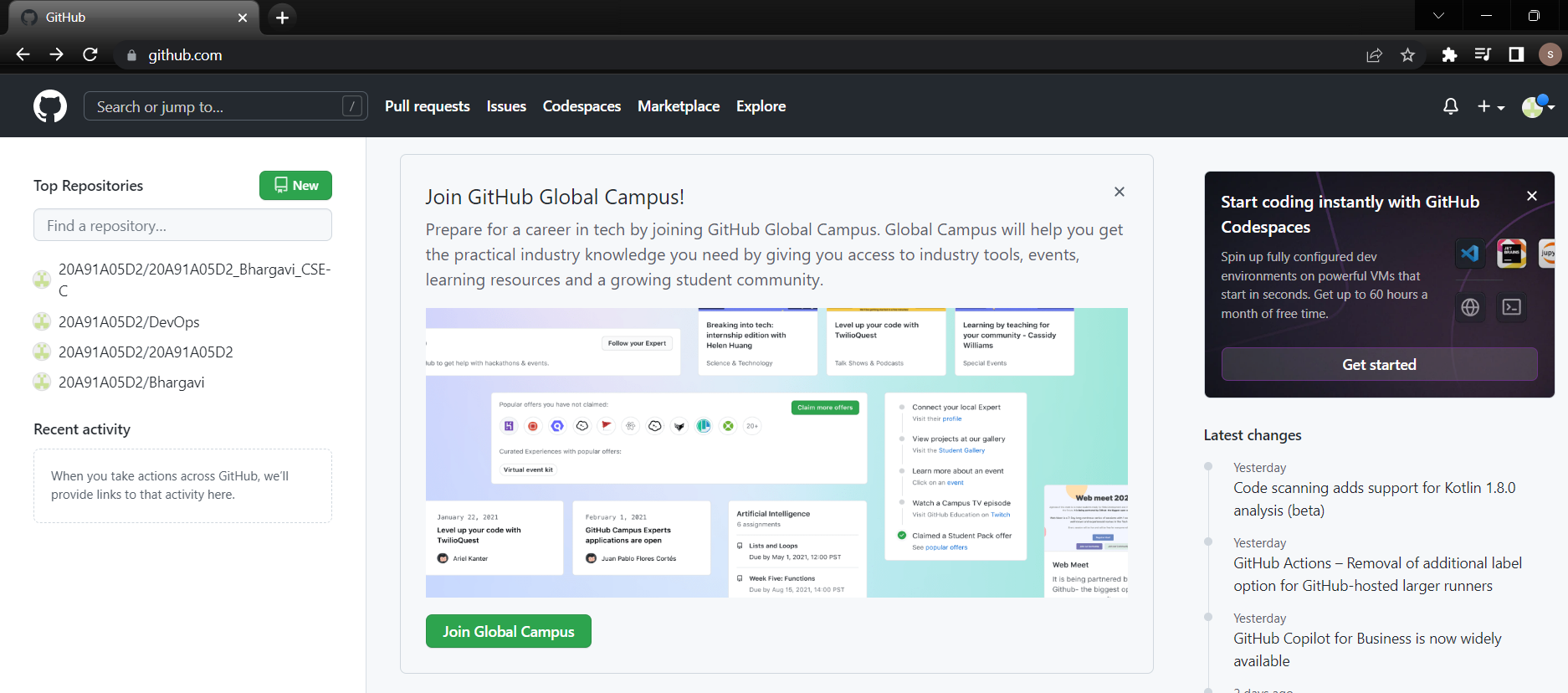
\***git stash clear** : Used to clear all the stashes from stash stack.



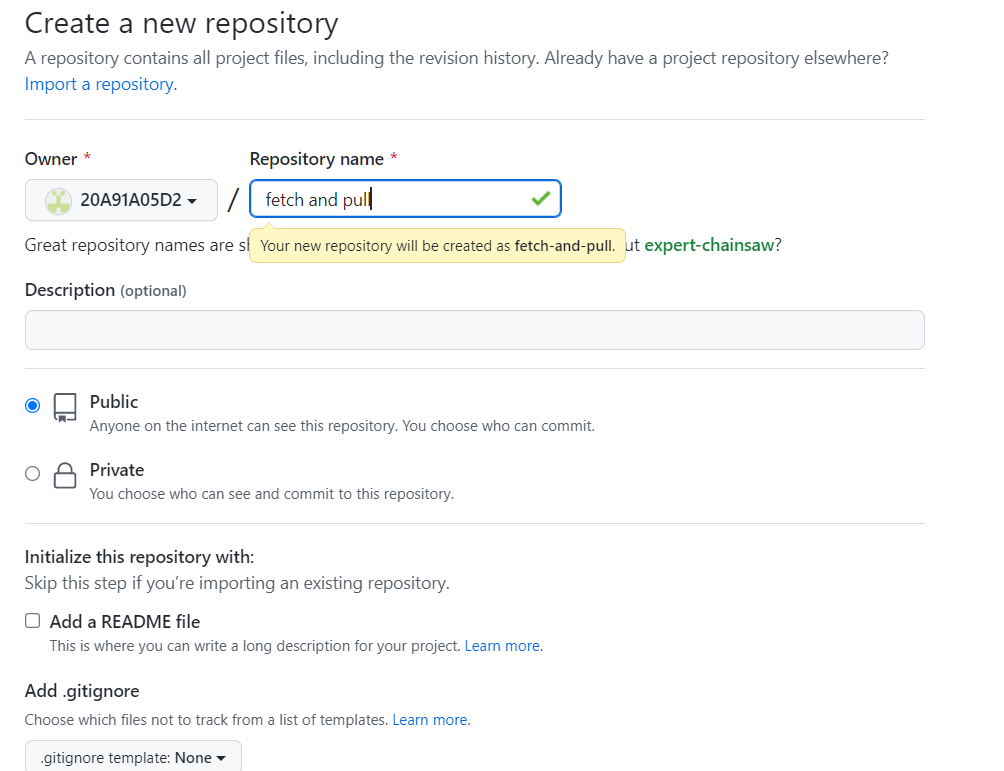
**Q2. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.**

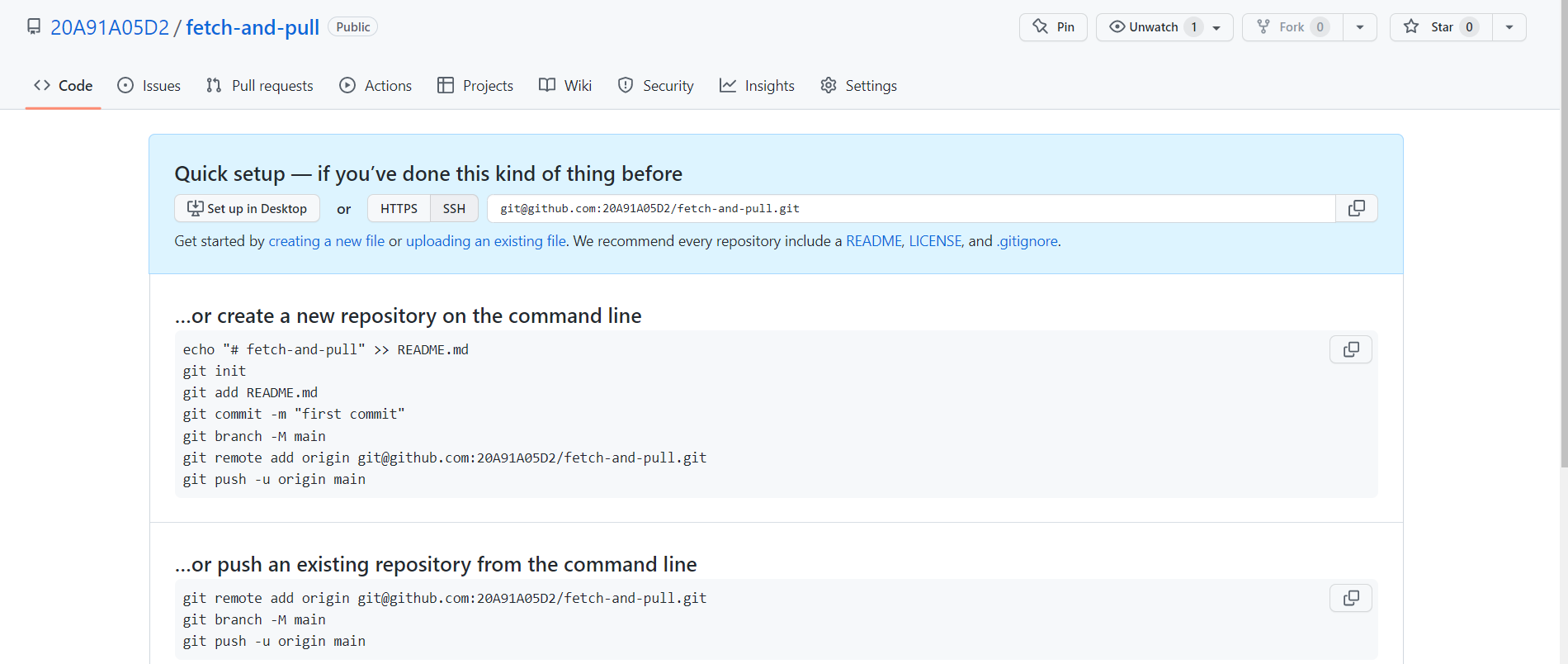
**Git fetch:** Git fetch is a command that allows you to download objects from remote repository but it doesn't integrate any of this new data into your working files.

\*Create a new repository in github.



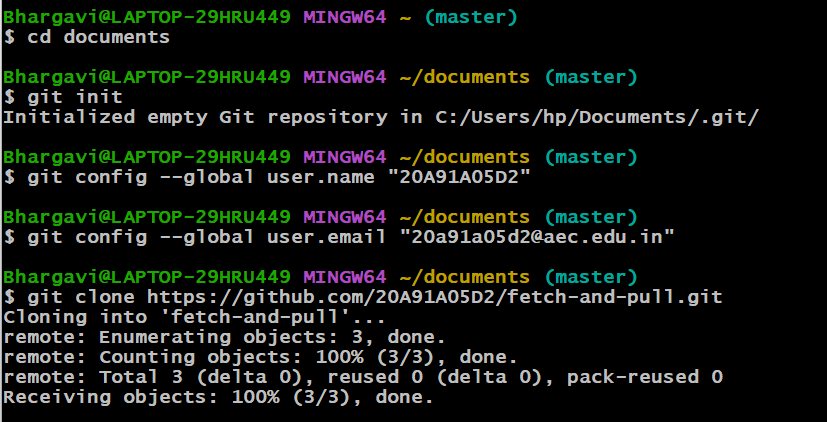




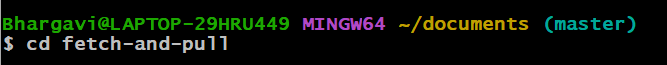


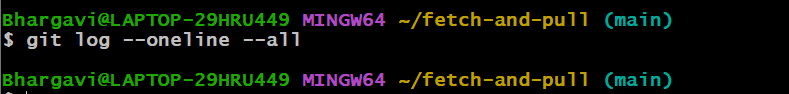
\*Clone the empty remote repository into local repository using git bash.



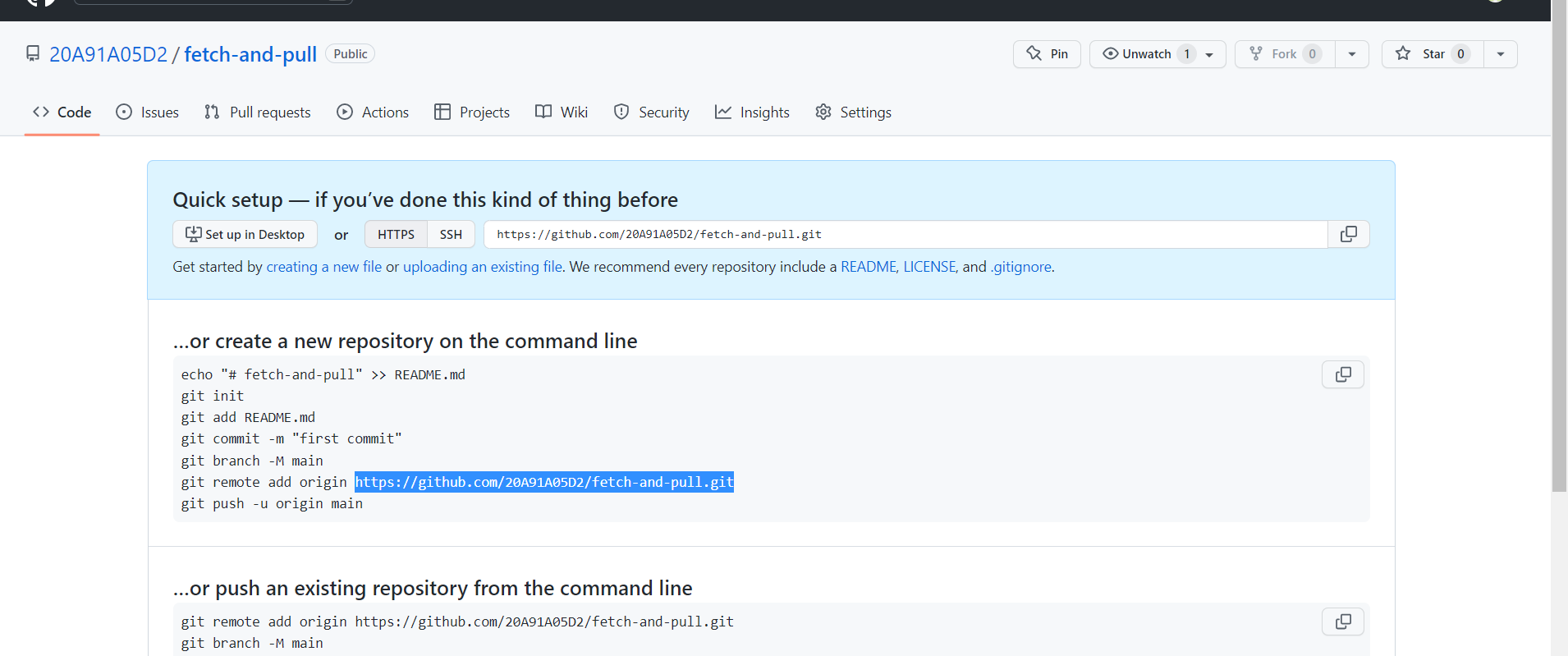


\*Check if there are any commits present in the repository.

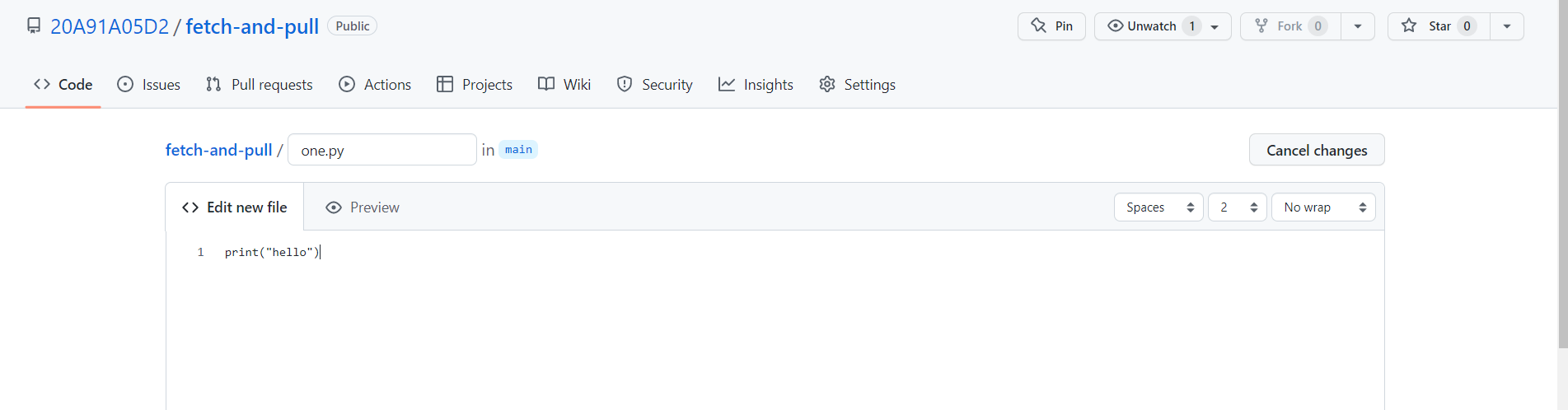


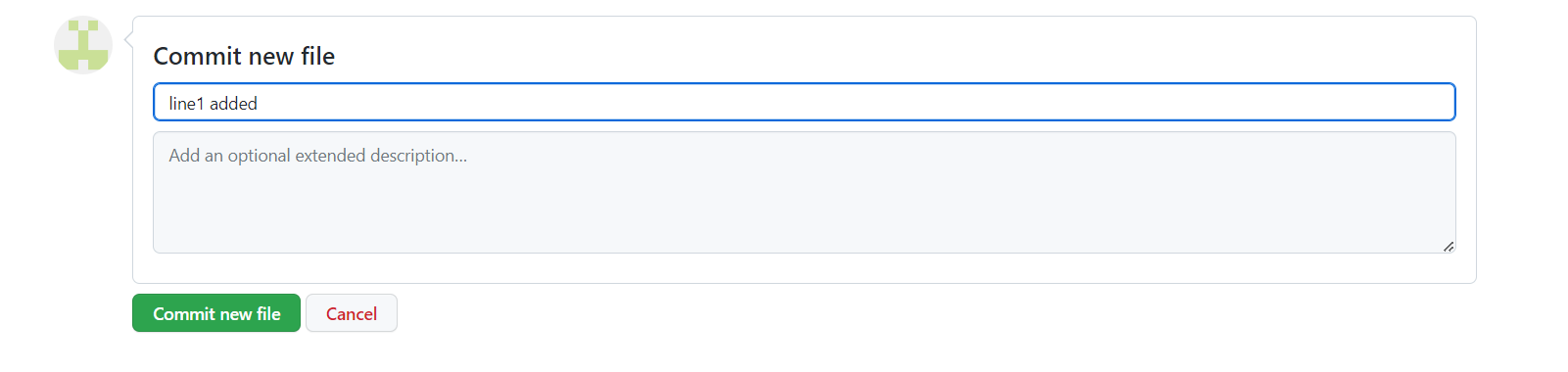


\*Create a file in the github.

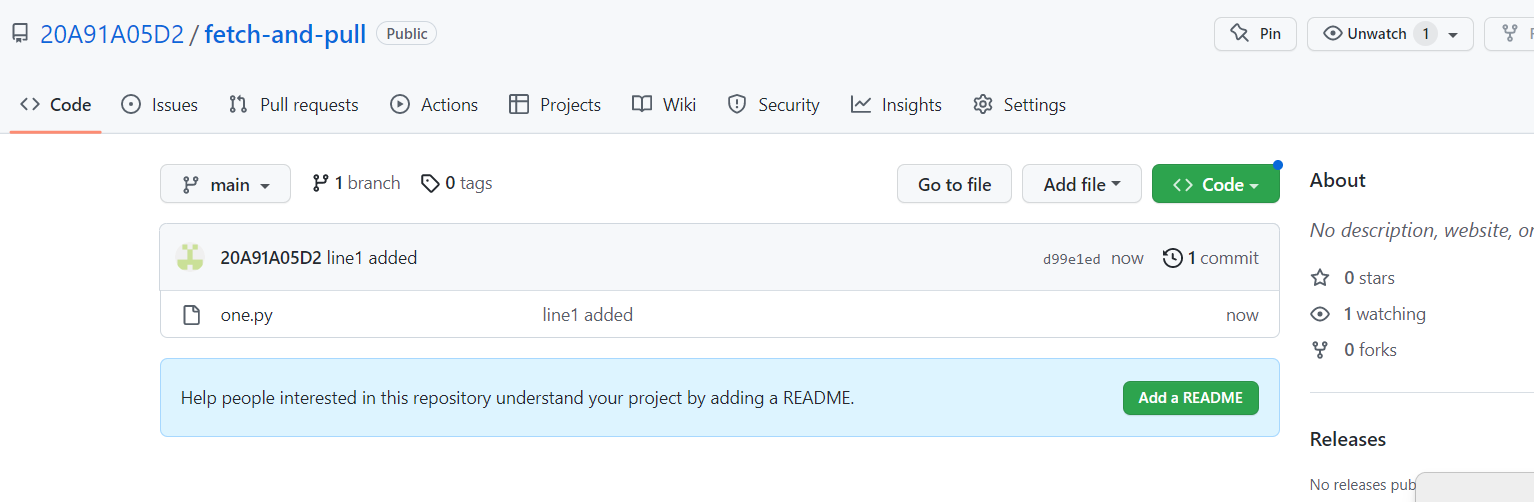


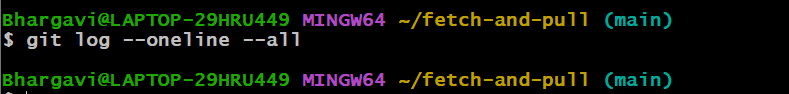








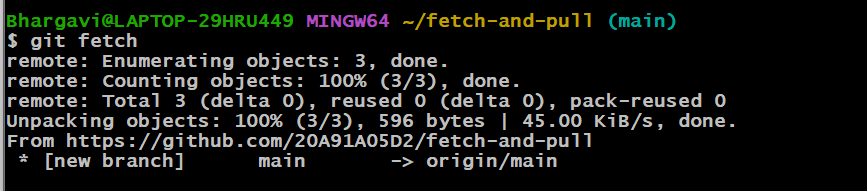




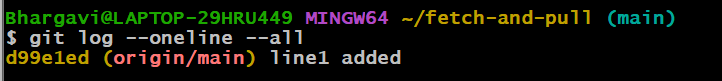
We cannot find any commits in our local repository even if we made a commit in remote repository.

\***Git fetch:**

Let us fetch the repository. It will download all the changes that are made in the remote repository but doesn’t merge our changes with working files.



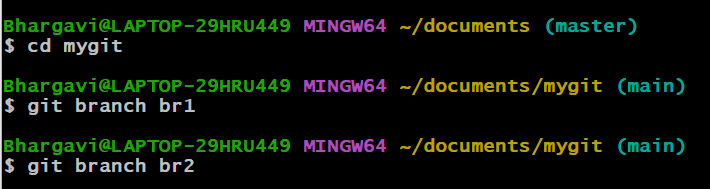
\*Use the git log --oneline --all command to check whether the changes have been downloaded or not.



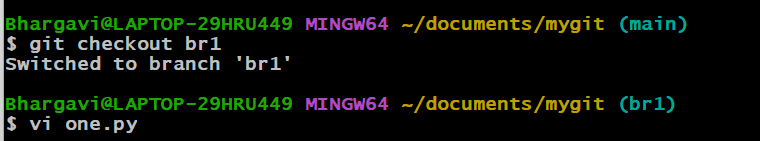
\*Here using git fetch the commits are not applied to the main branch.

**\*\*Git merge:** Git merging is basically to merge multiple sequences of commits, stored in multiple branches in a unified history or to be simple you can say in a single branch.

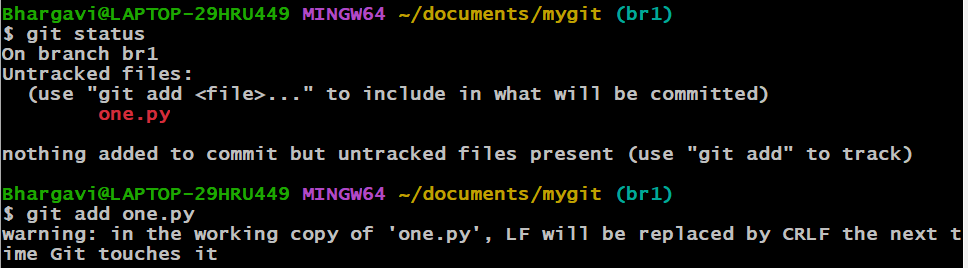
**\***Create 2 branches say br1,br2 using the command **git branch branchname.**

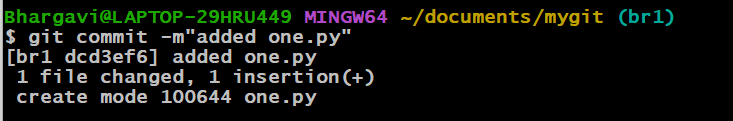


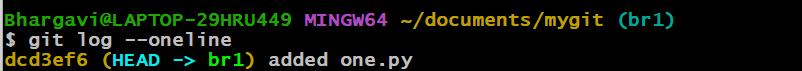
\*Switch to one of the branch for example br1 using the command git checkout branchname or git switch branchname and then create a file and commit the changes made in it.



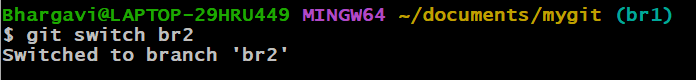






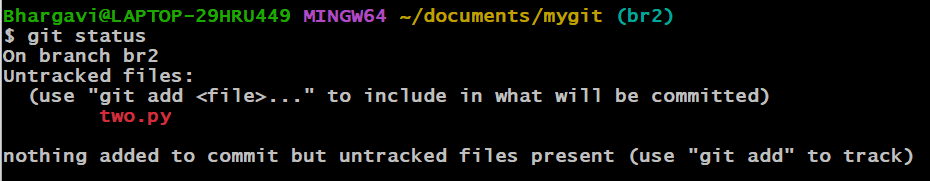


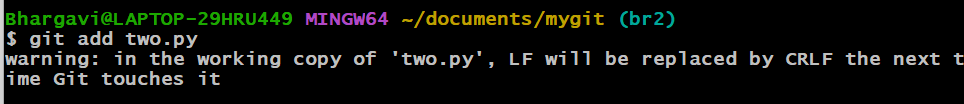
\*Now switch to another branch br2 and create and commit a file.

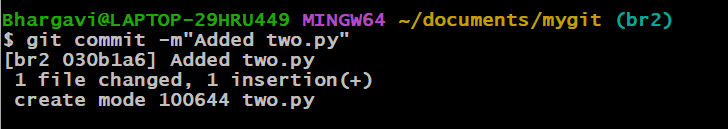








****

****

**\***There are two different branches whenever we need to combine the work of both the branches we use **git merge** command.

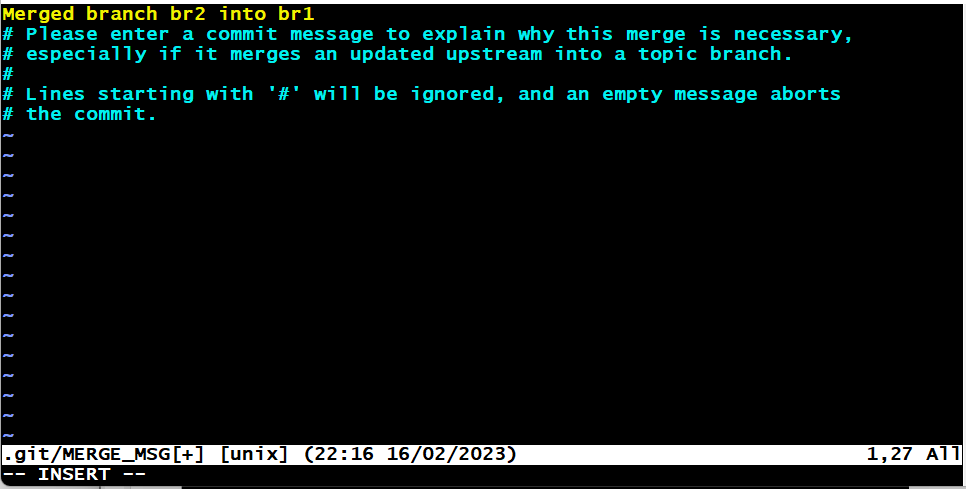
Syntax: git merge branch\_name

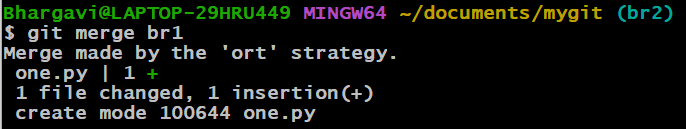
\*For example, let us merge the present branch we are in i.e, br2 with branch br1

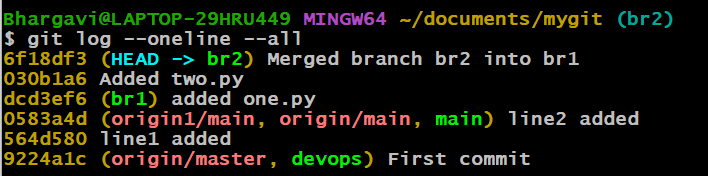


\*Whenever we made a merge then a new commit will be created which contains the changes from the latest commits of both the branches.

\*It will ask us to give a commit message if it is not given.



****

****

**All the commits that are made in the branch br1 are also reflected in branch br2.The new commit that was made during merging will be created in br2**

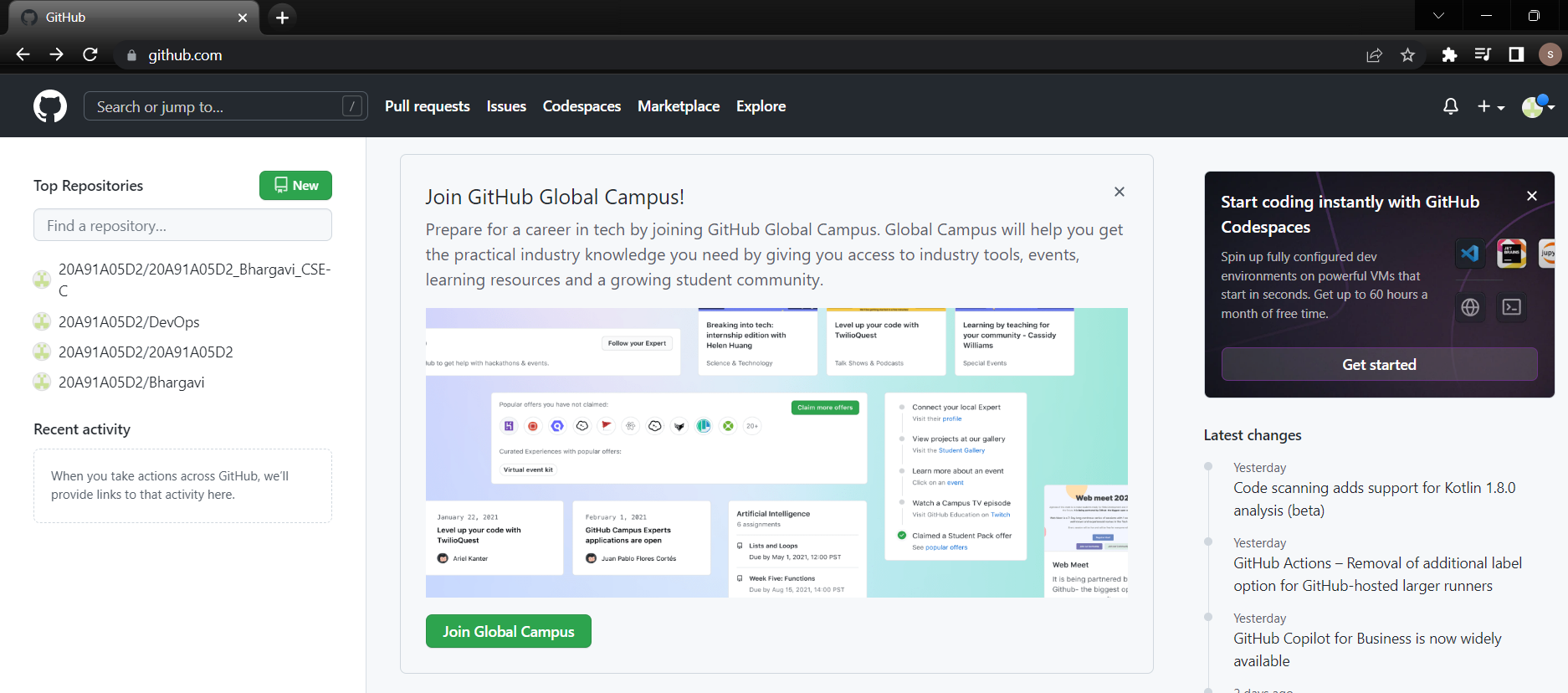
**Q3. State the difference between git fetch and git pull by doing a practical example in your git bash and attach a screenshot of all the processes.**

**Git fetch:** Git fetch is a command that allows you to download objects from remote repository but it doesn't integrate any of this new data into your working files.

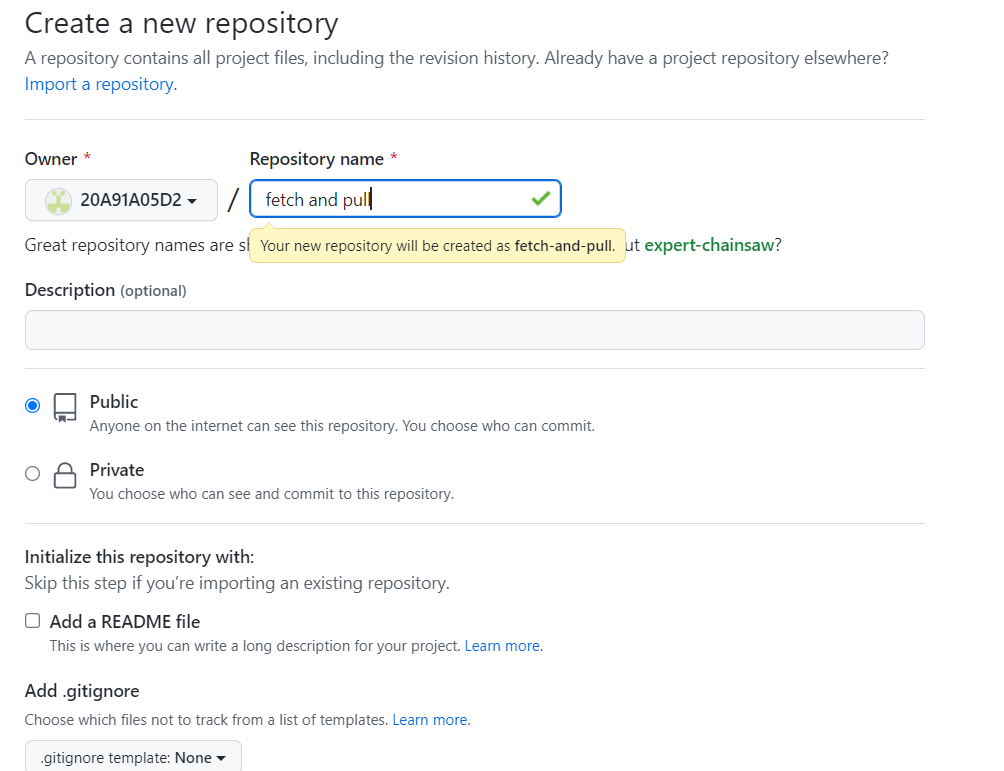
**Git pull:** Git pull is a command that allows you to fetch from and integrate with another repository or local branch. It update your current HEAD branch with the latest changes from the remote server.

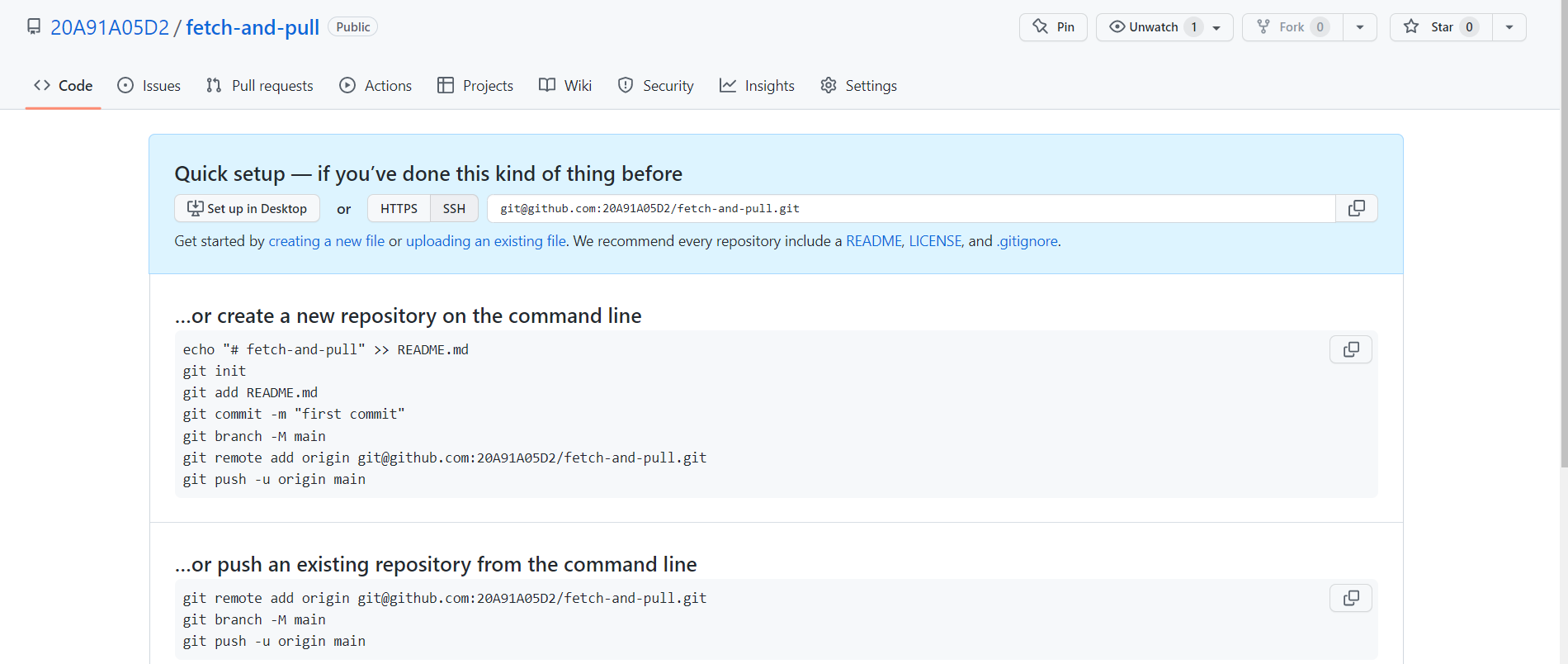
\*We can say that git pull is a git fetch followed by an additional action say git merge.

\*Create a new repository in github.



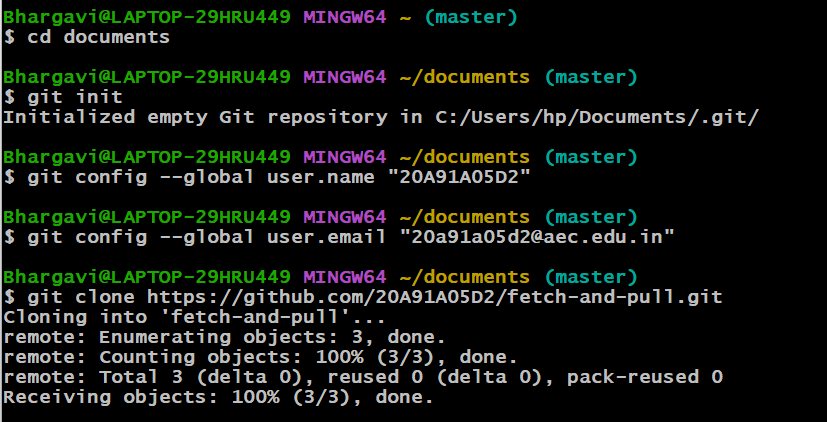




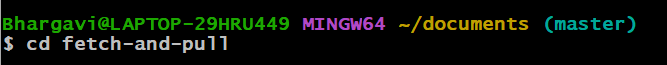


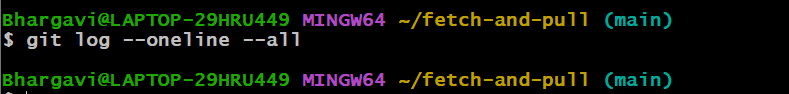
\*Clone the empty remote repository into local repository using git bash.



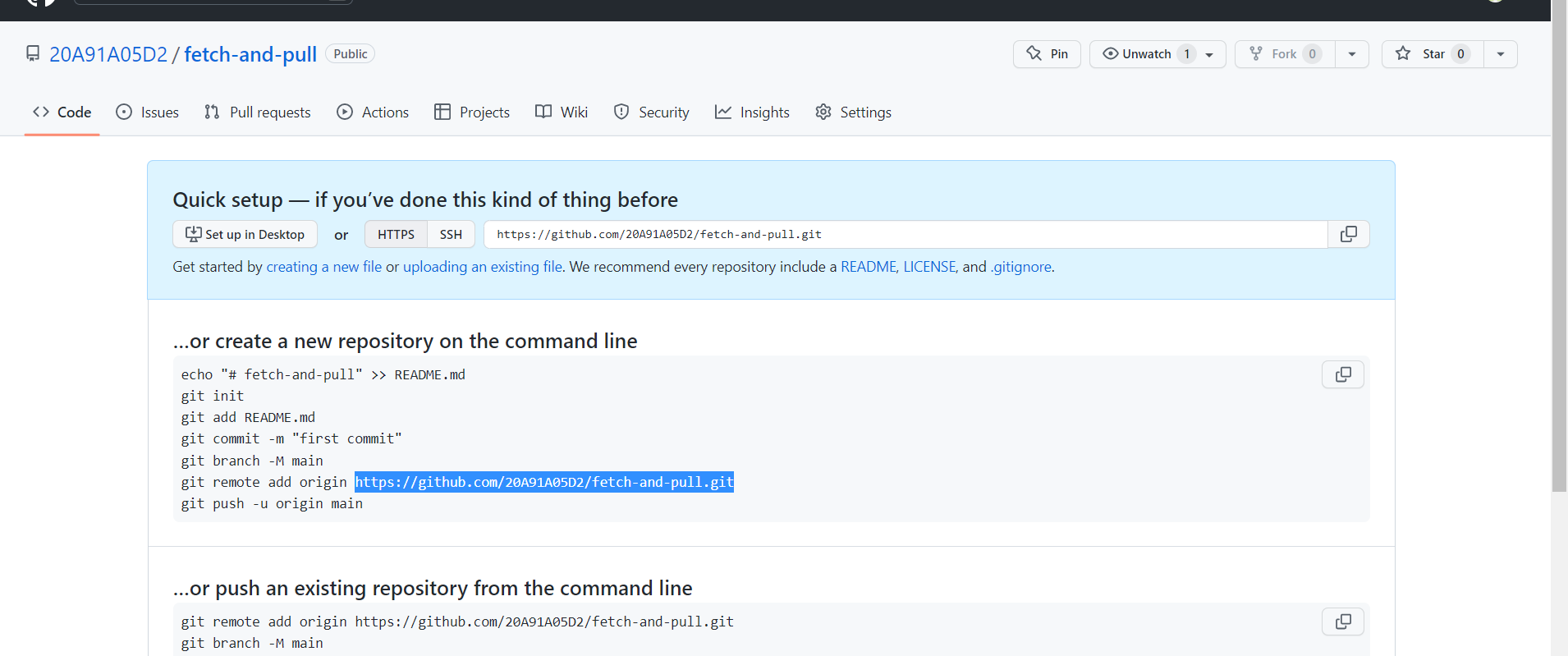


\*Check if there are any commits present in the repository.

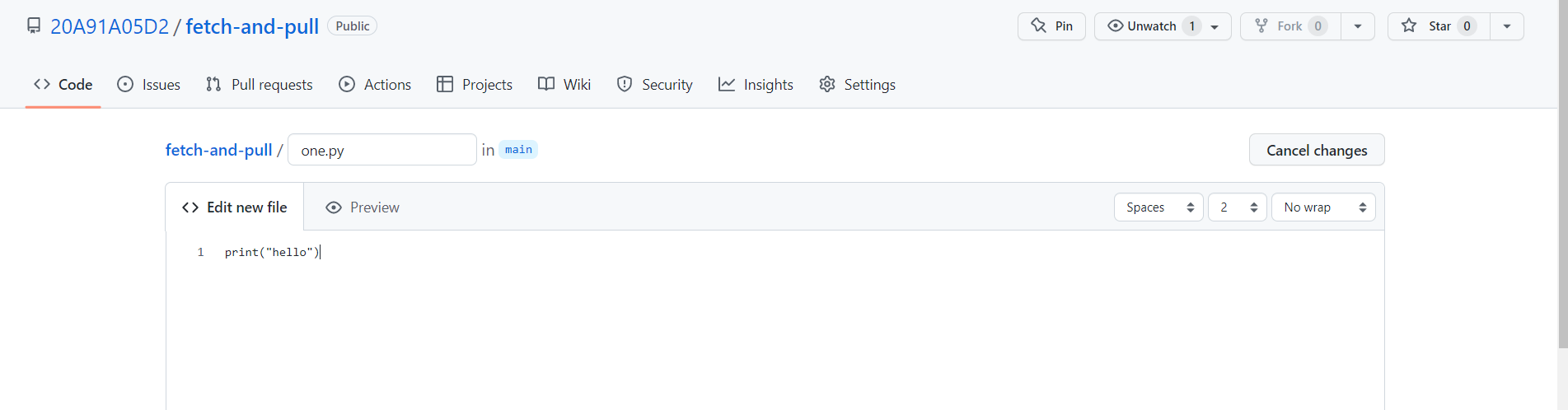
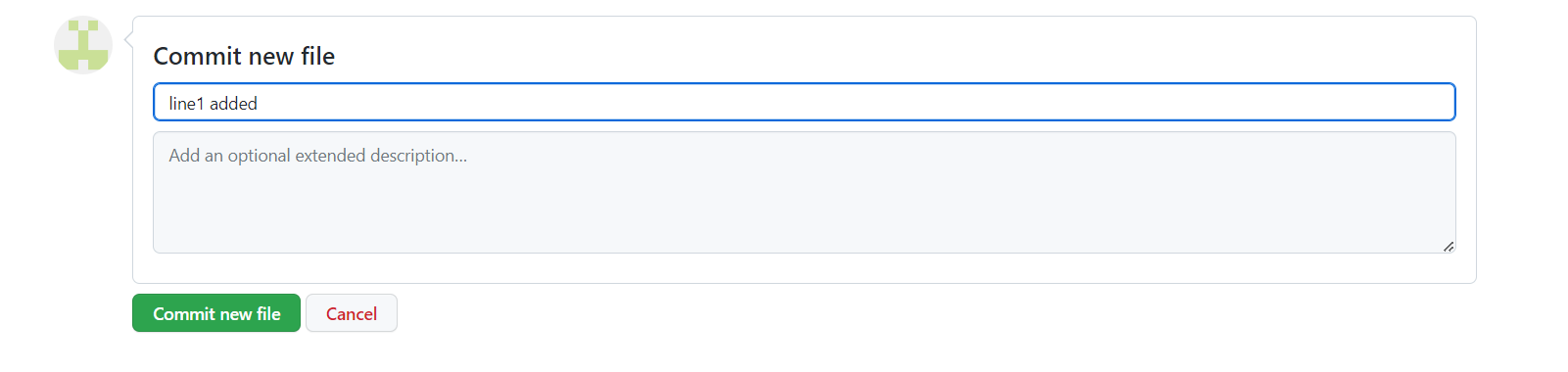


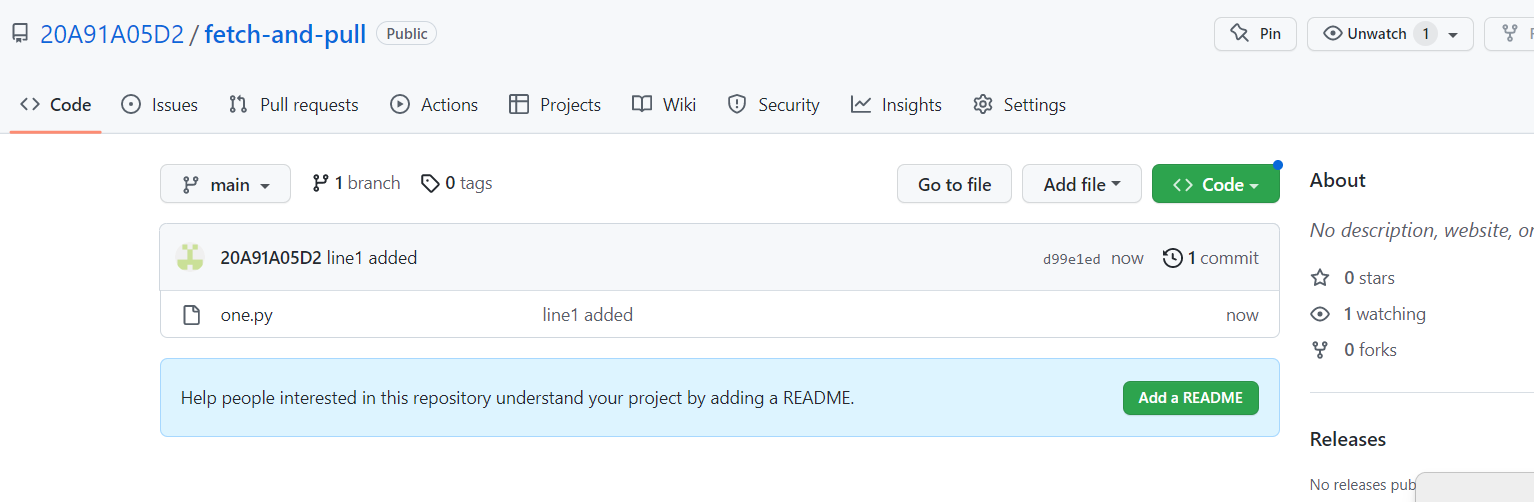


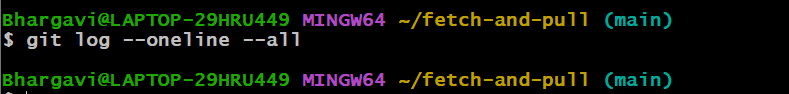
\*Create a file in the github.







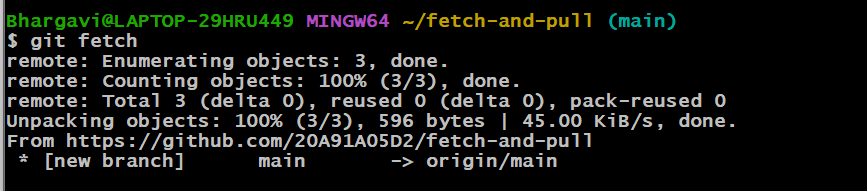




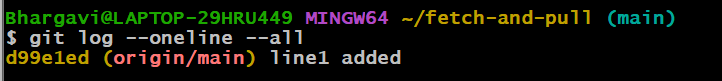
We cannot find any commits in our local repository even if we made a commit in remote repository.

\***Git fetch:**

Let us fetch the repository. It will download all the changes that are made in the remote repository but doesn’t merge our changes with working files.

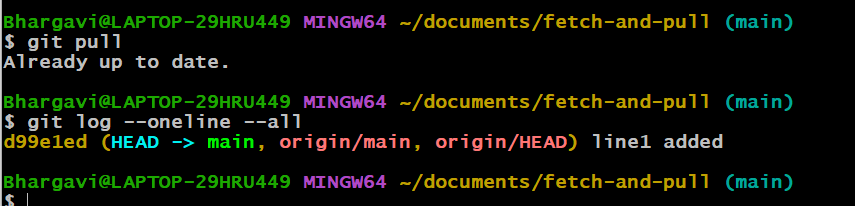


\*Use the git log --oneline --all command to check whether the changes have been downloaded or not.



\*Here using git fetch the commits are not applied to the main branch.

\***Git pull:** It downloads and also applies the changes to the working files.



Here the HEAD->main is present which indicates that the changes have been applied to the present branch.

