**Introduction to Git and GitHub**

**GIT**

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Using git we can tracking changes in our computer files. It is used for coordinating work among several people on a project and tracking progress over time.

**Git hub:**

GitHub makes it easier to work on projects together, share code, and manage different versions of your work.

**Config:**

It’s used to configure settings. allows you to customize how Git works and optimize it to fit your workflow

**--Global:**

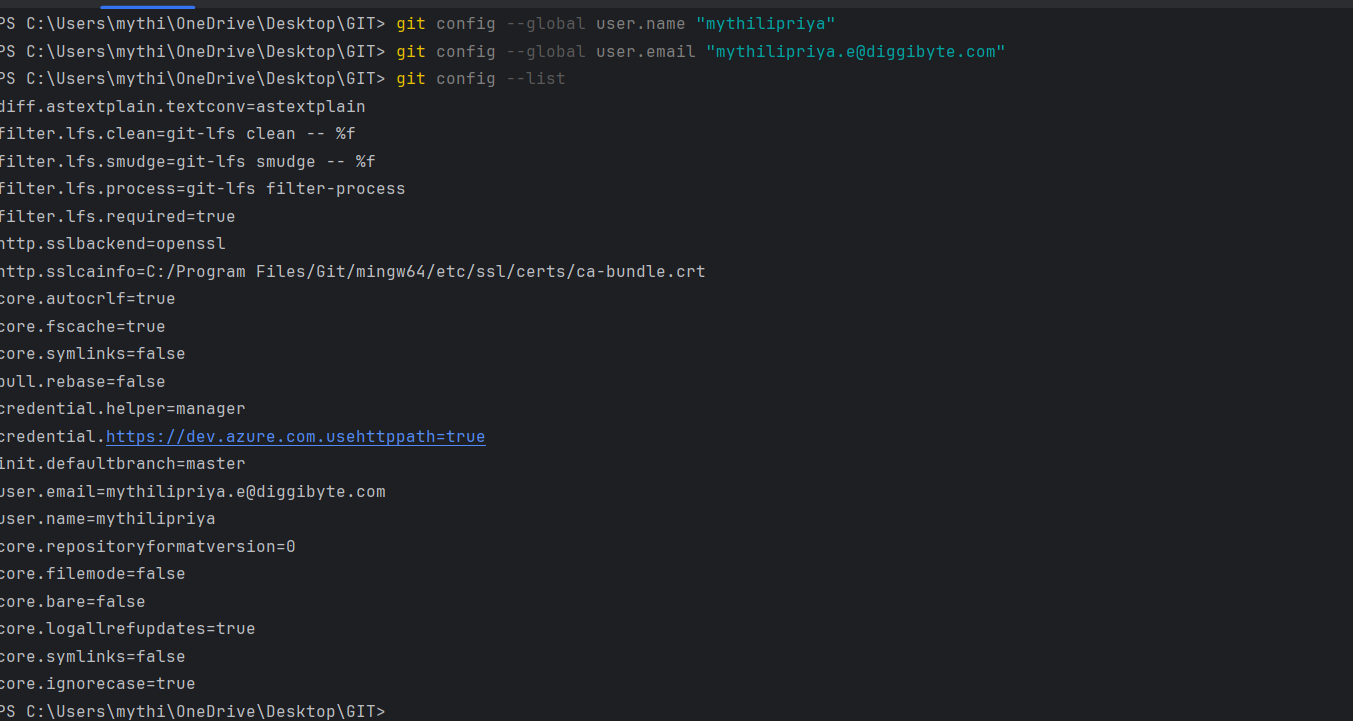
Without the --global option, this setting will only apply to a specific repository.

**User name and email:**

 This information is used to identify the author of the commits. You can set these values globally (for all repositories on your system) or locally (for a specific repository).

Name is probably the most *universal* way of identifying a human in the "real world", and email address in the "online world." So the combination of the two is a good way to identify the people who have contributed to a project

Name need “ “



**Commends:**



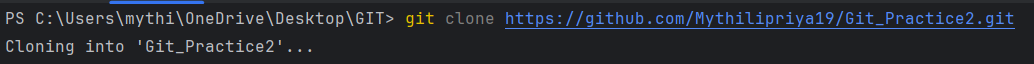
**Staging area:**

Temporary workspace

**Git clone:**

Here we creating a complete copy of a project from a remote repository (like GitHub) onto our local computer.

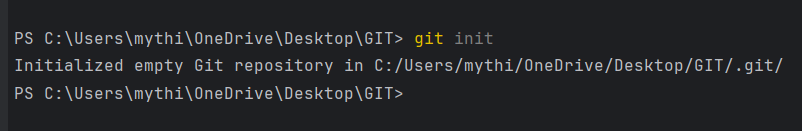
$git clone “ URL”



**Init:**

initialize a Git repository for our local project folder.

$ git init

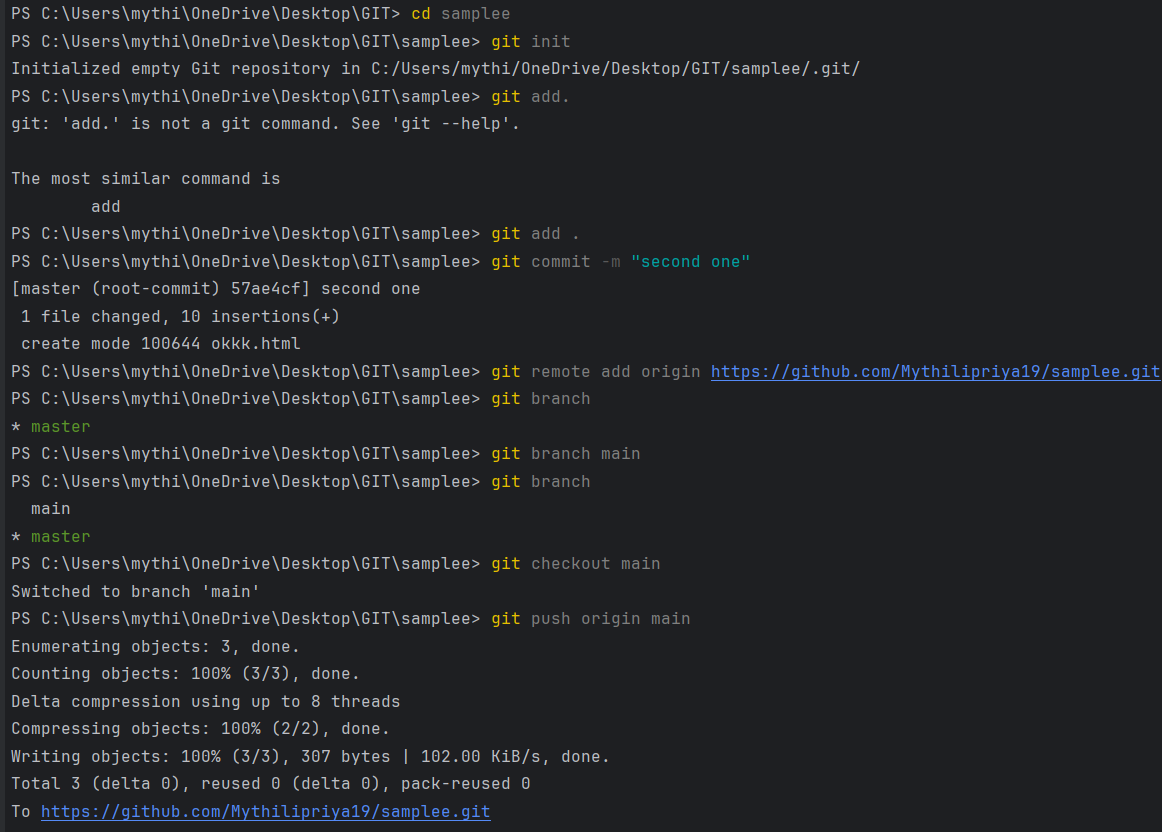


**Git add:**

Collects and prepares your changes for the next commit.

$ git add (particular file name)

$git add . (This will take all our files into the Git repository)



**Git commit:**

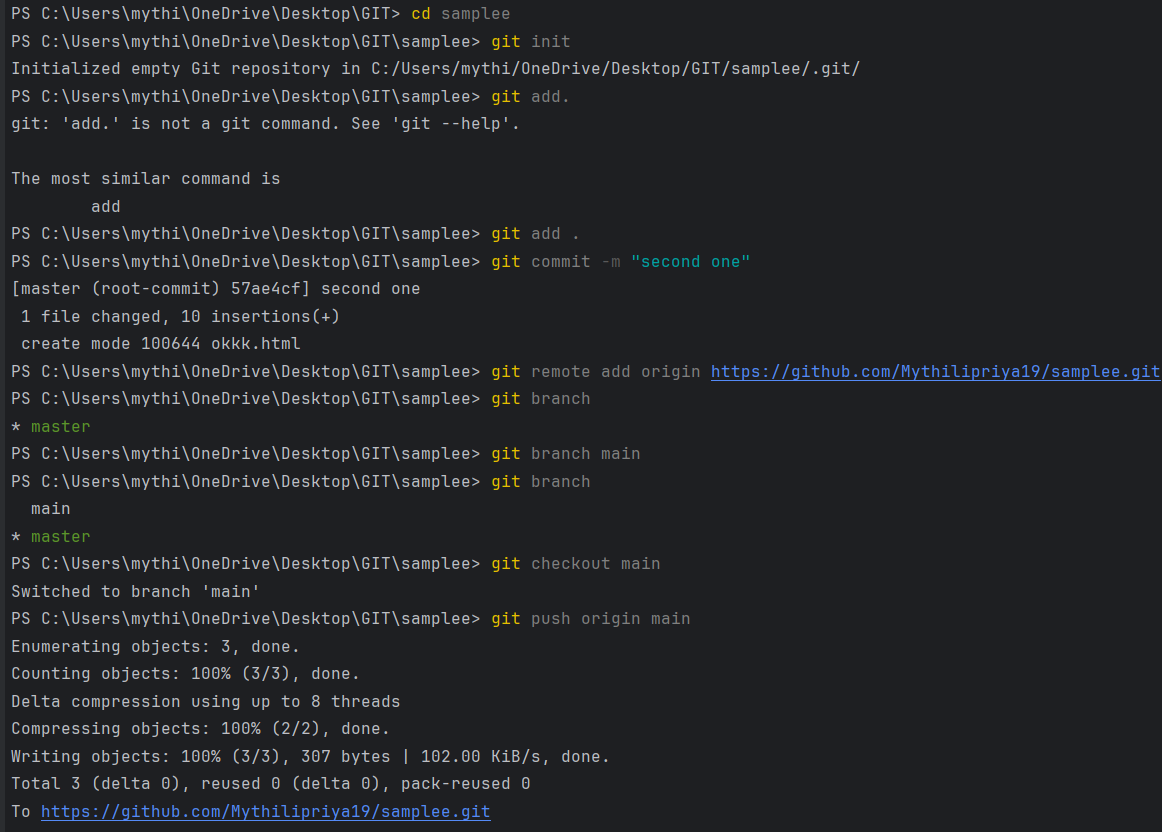
Saves the changes you’ve staged with git add and creates a permanent record of those changes in the project history**.**

used to move files from the staging area to your local repository

After executing the git commit, your staging area will be empty.

Simply to save files

git commit -m “commit message”



**Git status:**

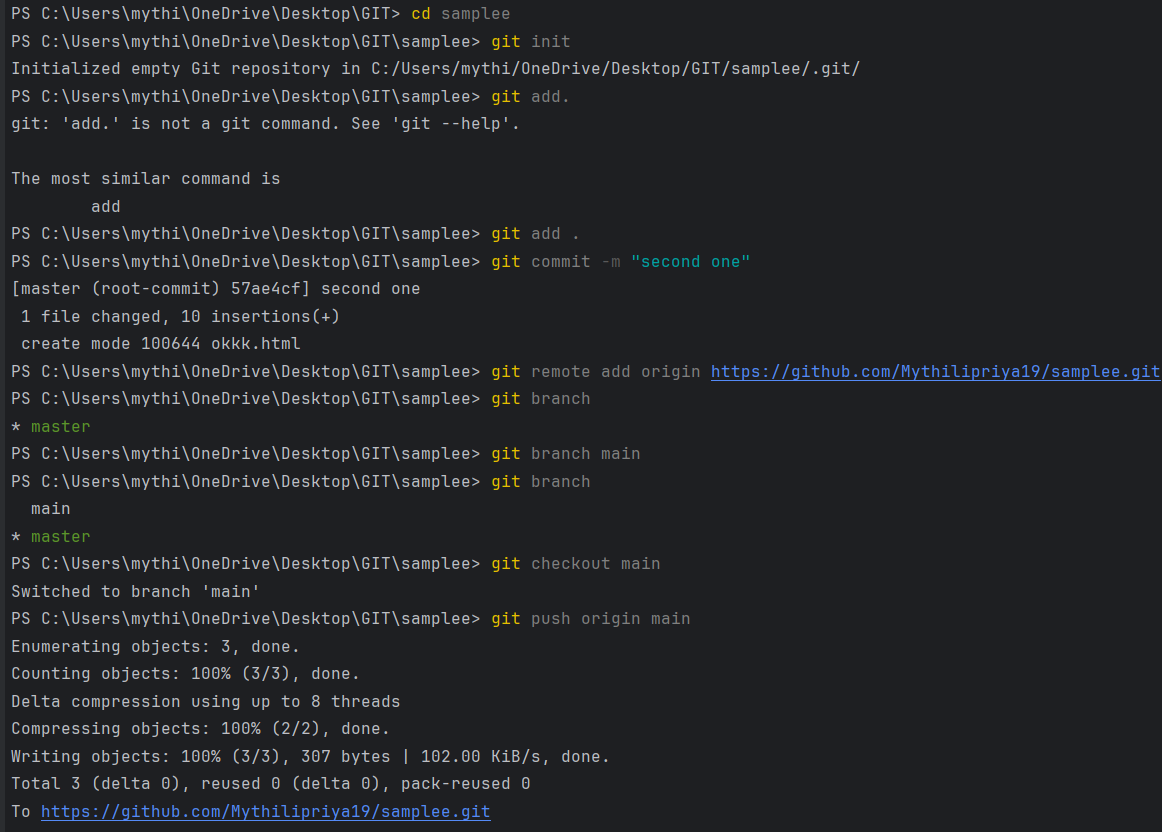
displays the state of the working directory and the staging area

$ git status

**Git push:**

Upload all the content from local repository to remote repository.

$git push



**Git pull:**

helps you stay up-to-date with the latest changes that others have made to the project. From remote repository into the working directory

**Git rm:**

Used to remove file

Git rm filename: deleted in both working directory and staging area

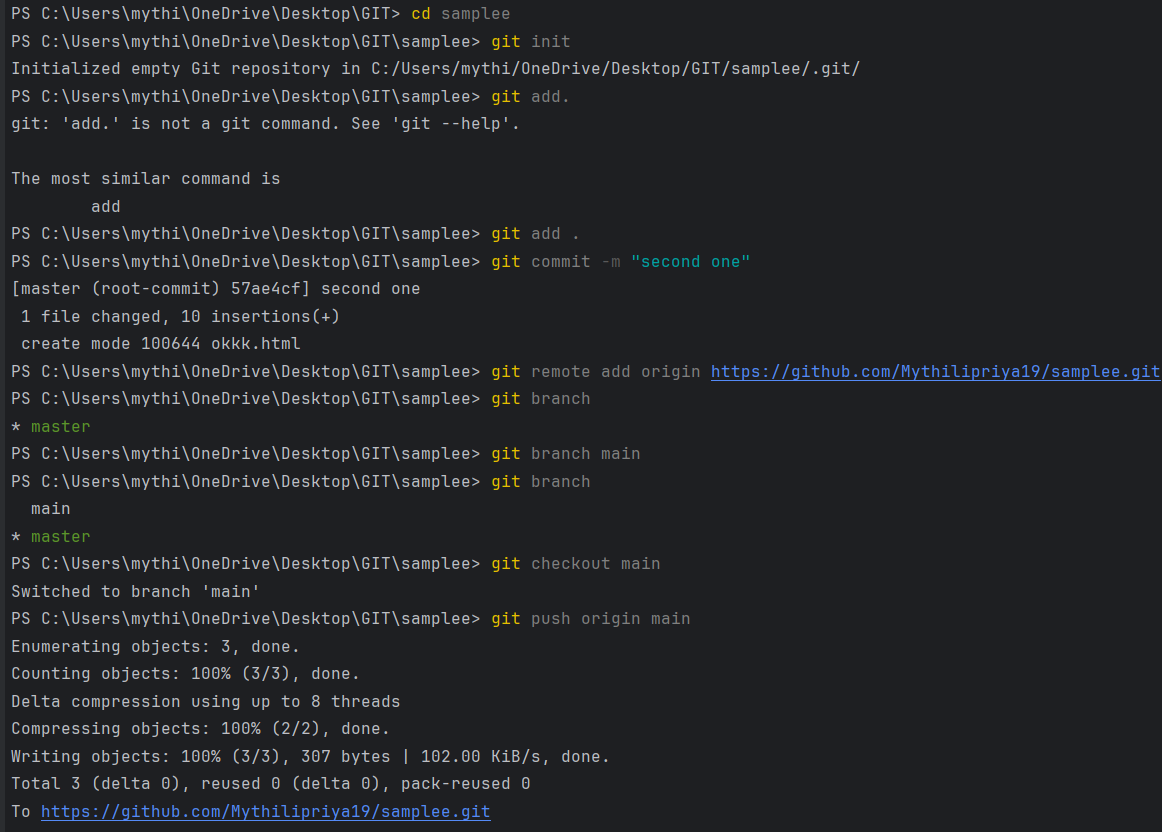
Git rm –cached file name: File deleted only in staging area

**git remote:**

 project that are hosted on the Internet or network somewhere.

 $git remote add origin “[URL]”

Once everything is ready on our local system, we can start pushing our code to the remote (central) repository of the project



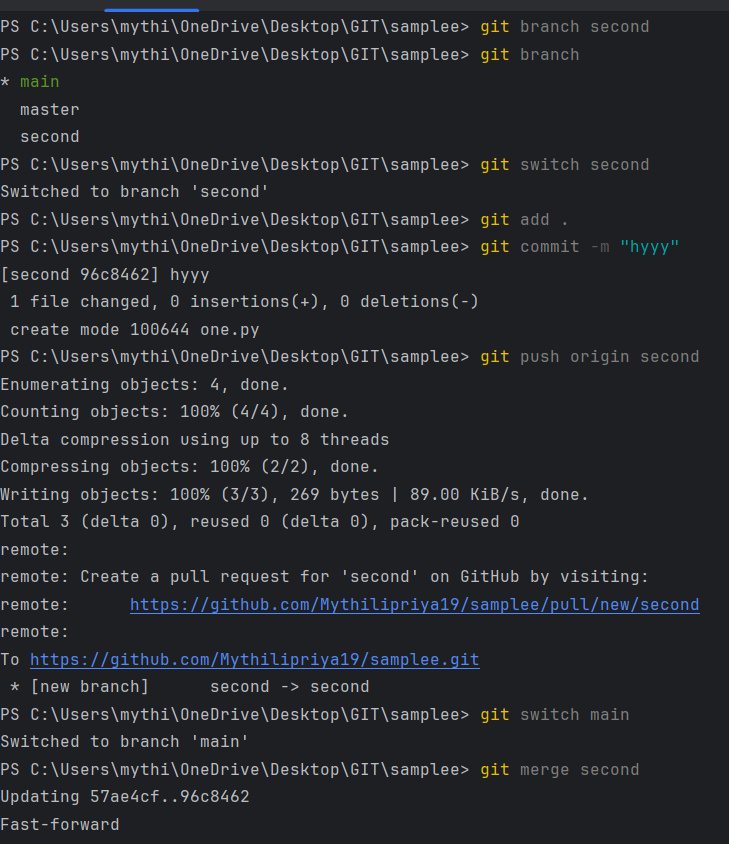
**Git branch:**

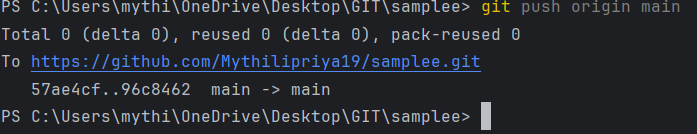
The git branch command can be used to create a new branch.

* Git branch : used to list out the branch names
* Git branch branch name: create new branch
* Git switch: used to switch one branch to another branch
* Git checkout branch name: switch
* Git checkout -b branch name: in this also we can create branch and that branch directly switched to newly created branch
* Git branch -m new name: for rename branch
* Git branch -d branch name: delete branch
* Git push origin –delete old branch name

**Merging:**

Pulls in the latest changes from main into the feature branches

 $Git merge branch name

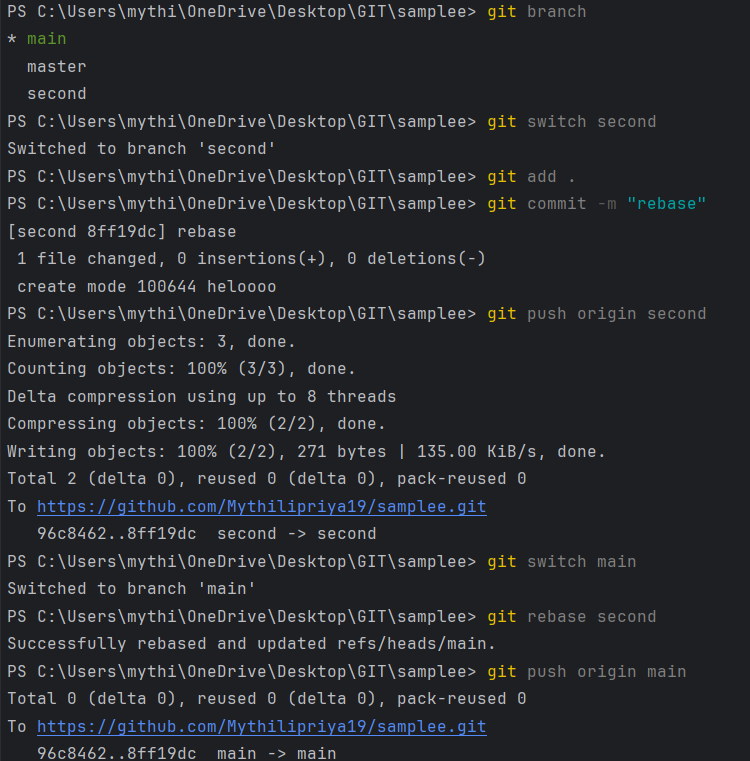


**Rebase:**

Moves or re-applies commits from one branch onto another branch, creating a linear commit history.

It gives clean,straightforward commit history

$Git rebase branch name



**Fetch:**

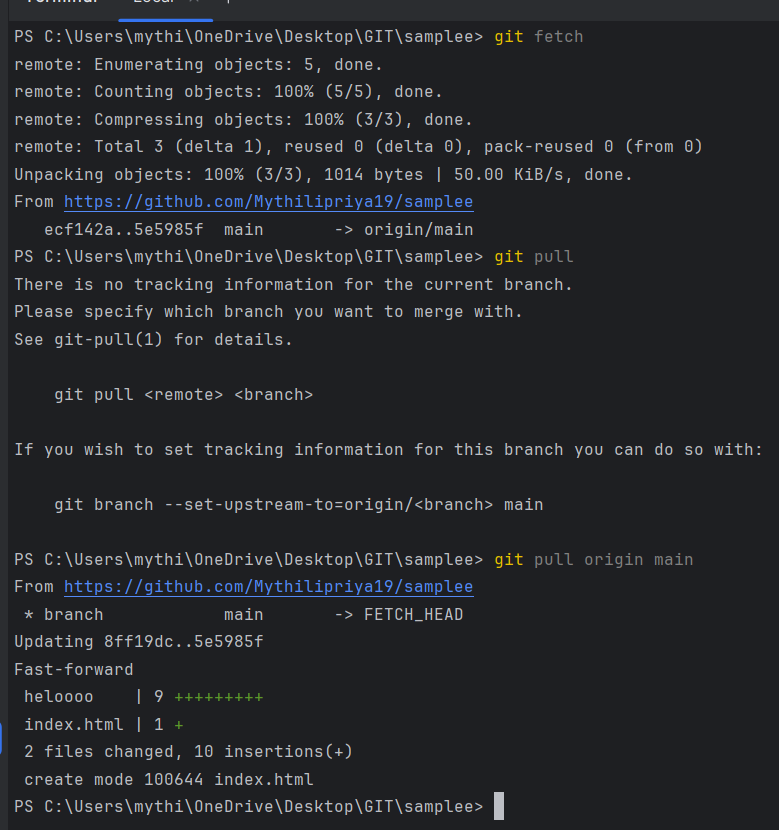
Its used to notify changes in remote repository

$Git fetch

**Pull:**

Directly download the changes from remote repository to local repository

$git pull origin main

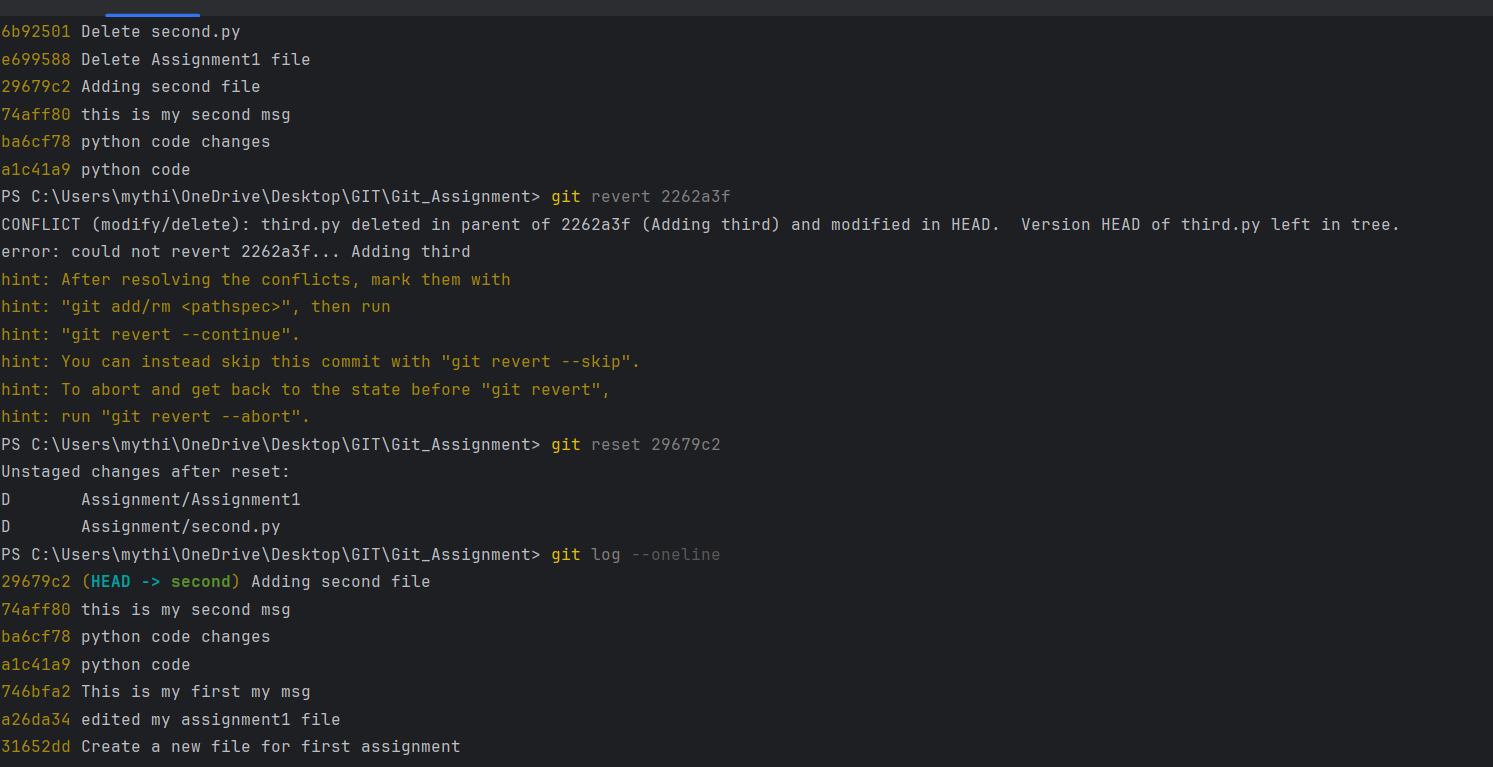


**Revert:**

Revert preserves the commit history by creating a new commit that undoes the change

Revert is a safer option when working with other developers because it doesn’t affect others’ work

**Reset:**

Reset rewrites the commit history by undoing the operations**.**

**Reset can cause other developers to lose history and have conflicts in their work.**

**Cherry-pick:**

**Pick the commit from one branch to another branch**

