**Git Overview:**

Git is an open-source distributed version control system

* Collaboration: Enables multiple developers to work on the same project simultaneously with seamless merging and conflict resolution.
* History Tracking: Records every change, allowing you to revert to previous versions if needed.

**Basic Commands:**

**Git init:** It is Used to convert local folder into git repository

$ git init

**Git clone**: Creating a copy of remote repository to local computer

$ git clone <Git-URL-LINK>

**Staging Area**: It is a Preparation Zone which is ready to be committed

**Git add**: It is Used to move to staging area

$ git add filename or **.**

**Git commit**: Save the changes to a repository

$ git commit -m “Commit Message”

**Git Push**: Moves a file from local to remote repository

$ git push origin main

Origin is a default name that git repository automatically assigns when we clone or add a file

**Git Status**: git status provides a summary of current state of git repository

$ git status

**Git log**: git log shows the commit history of our git repository

$ git log

**Git Pull**: Used to fetch the changes from remote repository into the working directory

$ git pull

**Git rm**: if we wrongly commit a file then using rm to remove a file

* Git rm file.txt – Removes a file from remote repository as well local repository
* Git rm --cached file.txt - If you want to keep the file in your working directory but remove it from the repository

**Branching**:

Git branch: Used to view the list of branches

Git branch new\_name: Used to create new branch

Git branch -m new\_name: Rename the branch - git push origin --delete old-branch-name

Git branch -d branch\_name: Delete a branch

Git switch branch\_name

**Checkout:**

**Git checkout -b new\_branch:** create new branch and checkout immediately

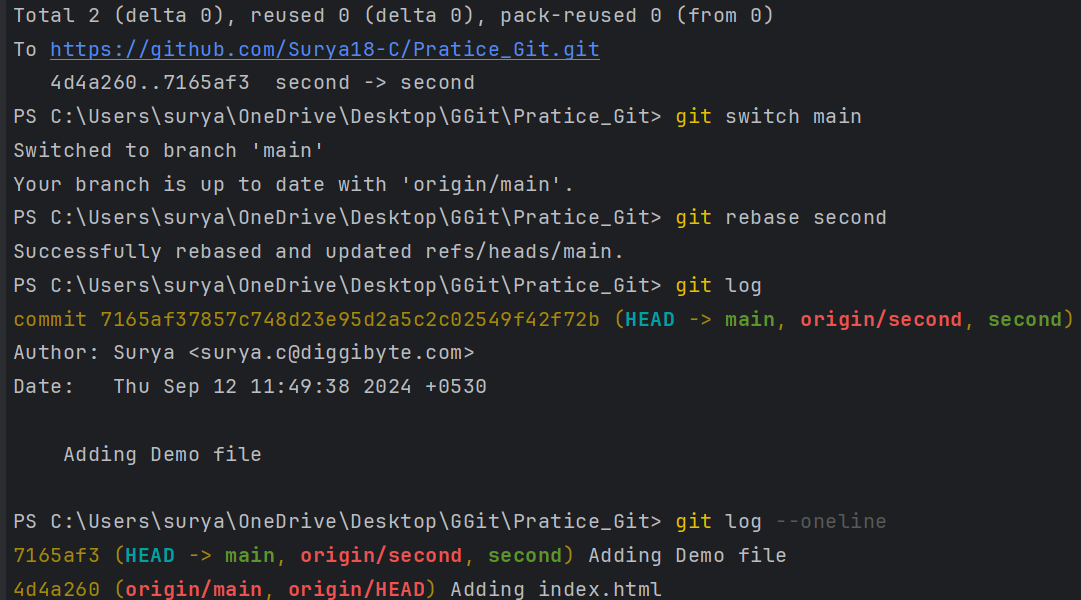
* + **git push origin --delete old-branch-name**

**Git Merge and rebase:**

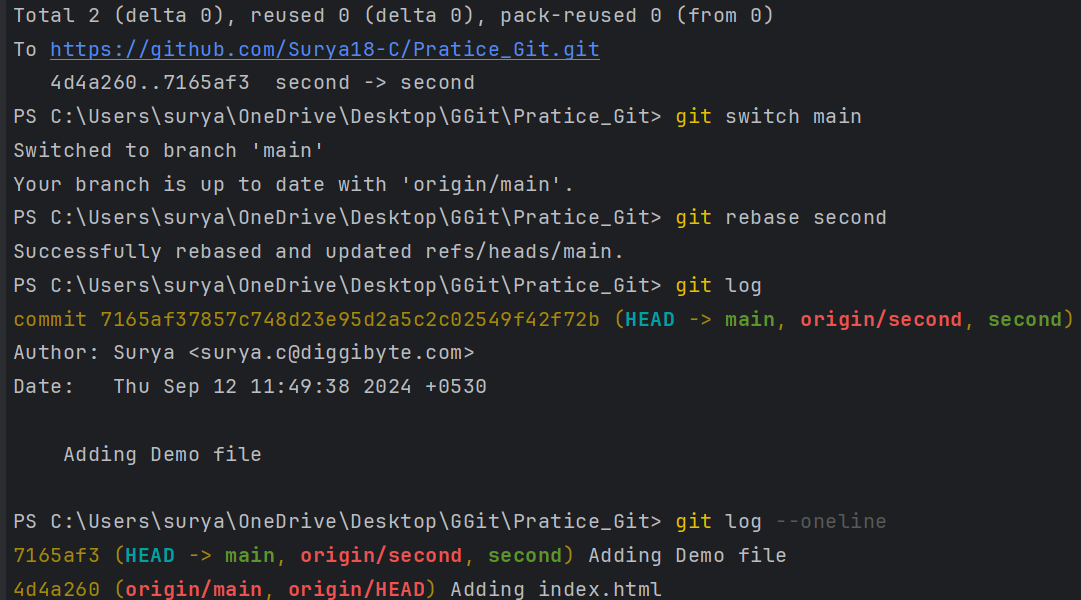
Git merge and rebase both are used to merging a one branch to another branch

**If we merge a content of future branch to main branch means then we use = git merge future branch**

* git rebase: Reapplies commits on top of another base branch, creating a linear history. Changes commit hashes as new commits are created.



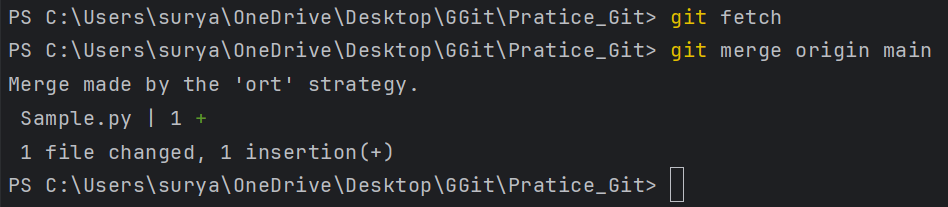
* git merge: Combines changes from different branches, maintaining the original commit history and creating a merge commit**.**

****

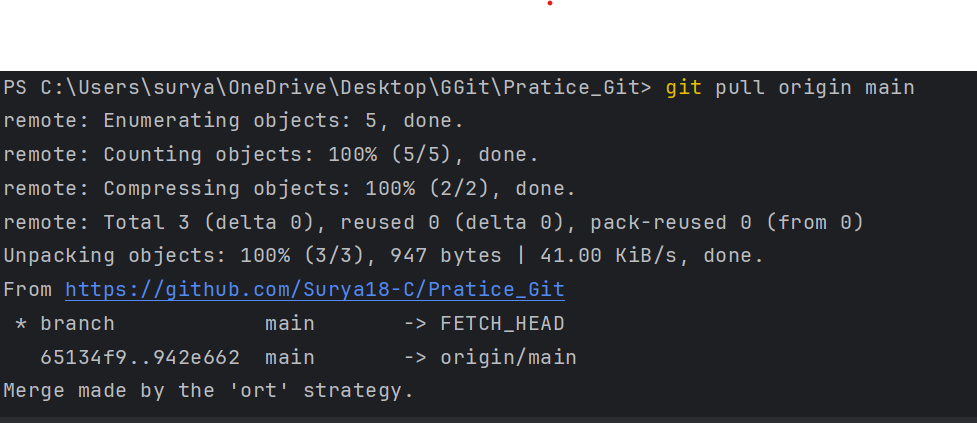
* **Best for Preserving History:** Use git merge if you want to maintain a detailed history and are dealing with multiple branches and merges**.**
* **Best for Clean History:** Use git rebase if you prefer a cleaner, linear history and are working on feature branches that haven’t been shared with others yet**.**

**Difference Between git fetch and git pull:**

* git fetch: Retrieves changes from a remote repository but does not apply them to your working directory. Requires a separate git merge to integrate changes – it just notifies the changes made in a remote repository.



* git pull: Fetches and merges changes from a remote repository into your current branch**.**

****

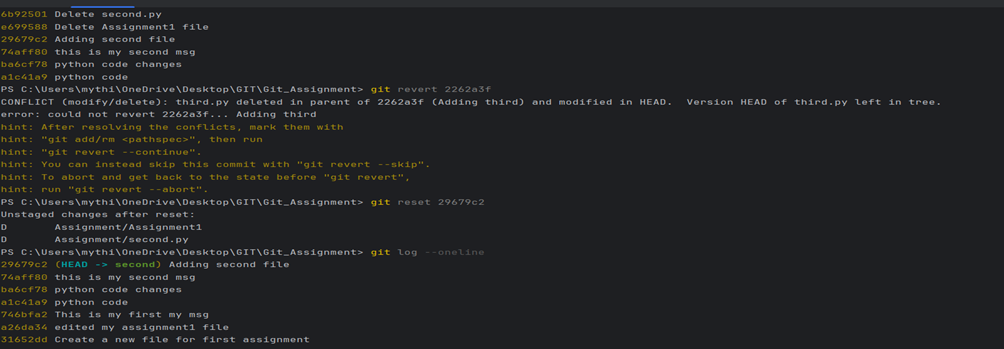
**Git Revert:**

* git revert <commit-hash>: Creates a new commit that undoes the changes made in the specified commit, preserving the history**.**

**Difference Between git revert and git reset:**

**Very first both are used to undo the particular commit**

* git revert: Undoes changes by creating a new commit that reverses the specified commit. The original commit and revert commit are preserved in history.
* git reset: **it is used to change the head position of commit** Moves the branch pointer to a specified commit, potentially discarding changes and affecting commit history based on the reset type (--soft, --mixed, --hard).



**Removing Git Repository:**

* rm -rf. git: Completely removes the. git directory and all Git tracking information from the repository**.**

**Merge Conflict:**

* Merge Conflict: Occurs when two developers make changes to the same file or lines in the same file and then try to push changes.
* Resolving Conflicts:
* Pull the latest changes.
* Manually resolve conflicts.
* Stage the resolved files.
* Commit the resolution.
* Push the resolved changes**.**

**Cherry-Pick Example:**

* if we accidentally commit changes to the wrong branch, you can use git cherry-pick to move that commit to the correct branch
* Example: If I have 2 branches one and two, I commit something in two but the thing need to place at one branch so in this time we use cherry-pick
* Git log --oneline = copy the commit hash in branch two Git switch one = go to branch one Git cherry-pick <commit-hash>
* Now the commit made in two branch is applied to one branch

