Git is a version control system to track the history of changes and to work collaborately.

Like in bank we have khata book.

**Commit (Change app pakka ho gya hai** ) =

* Git has main command “commit” it is like committing something like you commit in marriage that you cannot go back.
* History = whenever we do a commit that’s the history that git saves. So in short it saves commits inform of history.
* Change also mean screenshot in git so don’t be confused.
* Command = git commit -m “message”  
  Always write a short meaningful msg maybe you updated a feature so added a button or anything about the feature update, fix new bug etc.

**Add (Change abb add kar diya hai par abhi bhi change kiya ja sakta hai) =**

* And before commit there is adding the files to local repo of git, so add is like engagement in marriage so 1st add and then 2nd engagement.
* So, add adds new or changed file in your working directory to Git staging area.
* Staged files will be shown in green.

Remote = Folders and files that are on github

Local = Folders and files that are on PC

Command = git add <file name> or git add . (for all files)

**Clone Command**

Clone command is used to copy a github repo to our PC

Github PC/Laptop

(Remote) (Local)



git clone “link from git”

**status command** = To know the status of the code.

“git status”

**Types of status of git: -**

1. **Untracked** – Those files that are new maybe you made a new file in your local PC and git does not know about it. So, you must add and commit them first that’s what this msg is trying to tell you.
2. **Modified** – Means you have made some changes to a file and its pending to be staged(add) and commit.
3. **Unmodified** – No change in any files and everything is up to date.
4. **Staged** – When you perform an “add” command that means the file is added to the staged area and you must commit it to show it on remote github. \* Add vali stage ko hi staged kehte hai\*

Whenever you modify the file or do any change, we must do 2 things 1st 🡪 “add” and 2nd 🡪 “commit”

**PUSH –** The last step is to push your commits from local (PC) to Git (Remote).

Command = “git push origin main”

Origin is our default repo named by git (jha se hamne repo ko copy kiya tha laptop mai git usko bydefault origin repo bolta hai its name can be changed) where all our commits will be going to on github.

Main is our branch.

Command = “git push -u origin main”

Here -u is a upsteam mean that we are telling git that ham 1 hi project pe lambe time tak kaam karna chahte hai to baar baar origin main likhne ki zarurat nhi abb next time we can use git push only.

**Branches –** Sothat all developers can work at same time creating their own branches.

* Command to check which branch we are in – “git branch”
* Command to change the branch name – “git branch -M <branch-name>”
* Command to create a new branch and go into it – “git checkout -b <new branch name>”
* Command to checkout from 1 branch and go to another branch – “git checkout <branch name where u want to go>”
* Command to delete branch – “git branch -d <branch-Name>” (-d means delete)

A diagram of a project

Description automatically generated

**Merge branches: -**

Command to compare commits, branches, files etc. = “git diff <branch-Name>”

There are 2 steps to Merge branches 1 from terminal and other from Pull Request (PR).

1. Via terminal = “git merge <branch-Name>”
2. Pull Request = It’s like telling others about the change you have pushed so that they can review it before the merge.

Go to compare and pull request on github.

A screenshot of a computer

Description automatically generated



There you will see “able to merge” option in green

Let’s say your feature1 branch to your base branch that is main. Write a msg and create a new pull request

A screenshot of a computer

Description automatically generated

Click on Merge pull request and click on confirm merge.

A new commit will be added with a name Merge pull request from…

Because this is also a change so iska bhi screenshot liya ja rha hai. See below img.

A screenshot of a computer

Description automatically generated

PULL Command: -

Once we Merge the changes actually the changes are merged in remote that means git not in our local, so we have to pull the changes in our local as well.

So, pull is used to fetch and download content from remote repo to local repo and immediately update the local repo to match the content.

Command = git pull origin main

Resolving Merge Conflicts: -

What if 1 dev changes code in a line in his branch and 2nd dev changes code in same line in his separate branch and then if they try to merge then GIT will be confused that with line code is correct and then u will get a CONFLICT msg. Becoz GIT doesn’t know whose commit is correct.

Command = “git merge <other branch-name>”

Undoing Changes: -

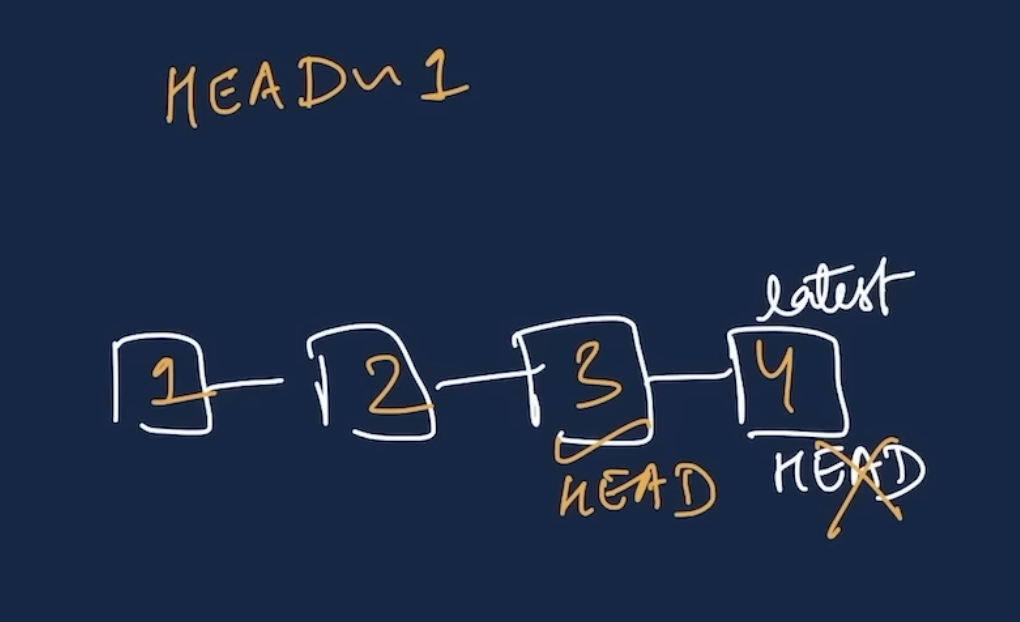
So, there are 3 scenarios: -

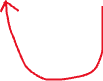
1. Staged changes – When you have added the changes to the staging area and how to check that. When u write git status it shows changes in green that means ready to commit.

Command = “git reset <filename>” (if only 1-2 files have changes to be reverted) or

“git reset”

1. Committed the changes (but only for one/single commit) – All the commits are stored internally in the git u can think it as a form of queue. The last commit name is always head, so head is the pointer pointing towards the last commit.





Command = “git reset HEAD~1”

HEAD~1 means HEAD to reset HEAD to 1 step back

1. Committed changes (for many commits) – That means we need to move HEAD pointer to some previous commit of our choice.

So, 1st we need to know that every commit has a “hash” id with it so if we want to go to some previous commit, we should know the hash ID of that commit.

A diagram of a code

Description automatically generated

From where to get the Hash ID???  
ANS – Using git log. See below pic

A screen shot of a computer

Description automatically generated



Command = “git reset <commit hash ID>”  
  
Copy the Hash ID of that commit and paste in the command.

Command = “git reset –-hard <commit hash ID>” (For VS code)

Using –-hard will remove the commit from VS code also.

Log – To check all the commits/logs/history.

Command = “git log”

Fork – Fork is like a rough copy of the upstream/main repository/project it shares same code and visibility settings with the main repo.

So, in which scenarios Fork is used??

ANS- Mainly when we want to contribute to some project, we Fork their project into our github account and do the changes in the code and then make a PR (Pull Request).

So, Fork is mainly used for contribution on github.

For Fork go to github page of the project and simply click on Fork button.

**Sometimes we start the project from local PC in that case how to initialize git from our PC let’s see: -**

**Step 1** = To initialize git in the folder we are working.

Command = git init

**Step 2** = add your files using add and commit them

**Step 3** = Go to GITHUB and make a repository

**Step4** = Now we have to link the local repo to the remote (Git) repo using the command below.

Command = git remote add origin <link of remote repo>

Notes\*

1. By default, the branch is main now earlier it used to be master but now it is main.
2. Just to know a folder is a git folder or not use ls -a to see hidden folders also there u will see .git folder that is what init has initialized for you in your pc.
3. Although it is seen that 1st make the repo on github and then clone in your local pc it is comparatively easy.