**What is GitHub?**

- **GitHub** is a cloud platform where you can **store your code** and **collaborate** with others.
- It is built on **Git** (the version control system).
- Think of it as a **remote backup + teamwork tool** for your Git repositories

**Git vs GitHub**

- **Git** = Tool to track changes in code on your local computer.
- **GitHub** = Website to host Git repositories online and share them.

**What is GIT?**

Git is a distributed version control system.
Means,

- It is a system that records the changes done to our files over time.
- We can recall any specific version of those files at any given time
- Many people can easily collaborate on a project and have their own version of the project files on their computer
- It is a online service that hosts our project

**Why GIT?**

- It stores the all the revisions of a project in just one directory
- We can rewind to any revision in the project
- Works on new features without messing up the main codebase
- Easily collaborate with other programmers
- Share our code with our teammates
- They can re-upload their edits and merger them with the main codebase. All the changes will be recorded

First, we have to install git

Go to google → download git ( https://git-scm.com/downloads ).
Depending on the OS, we should download the git

After downloading git

Go to command prompt → **git --version**
The above command will give the version of the git. It also indicates that, git is properly installed

ADDING USERNAME AND PASSWORD
To add the username → **git config --global user.name "username"**
To add the email → **git config --global user.email "email"**

To check the username and email

➔ **git config user.name**
➔ **git config user.email**
The above command will display the username and email

Few basic command lines

* To change the directory → cd ..
  This command will take us to the previous folder
* To change the location in command prompt → cd absolute_path
* dir → This will give the list of all the files and folders present in the location

* Suppose if we want to create a new directory in the location
  In command prompt → **mkdir dirname**

* To remove the directory
  In command prompt → **rmdir dirname**

* Suppose if we are using MAC or commander, to create a new directory
  In command prompt → **touch dirname**

**REPOSITORY**

* They are the containers for the projects that we want to track with git
* Repository can be locally stored on our computer or remote which is store on some kind of online service like GitHub
* We can have any number of repositories for different projects and all of these repositories are going to be tracked independently of each other by git.

There are three different stages

| **MODIFIED** | **STAGING** | **COMMIT** |
|---|---|---|
| Changed files, but not committed | Add the changed files to the staging area that we want to commit | Any files in the staging area are added to the commit. |

* Suppose we want to make some changes to the file; those changes will go into the modified stage. They are modified but not ready for the commit.
* If we decide to commit, we should add that file to the staging area.
* When we commit those files are going to be committed and are ready to upload to GitHub.

**NOTE** : **We can commit the files which are present only in the staging area**

**LET US CREATE A NEW LOCAL REPOSITORY**

* Take the path of the project where we should create the git repository
* Go to command prompt → cd project_location
* In command prompt → **git init**
  This will initialize empty git repository.

**STAGING AREA**

* In command prompt → **git status**
  This will give which files are modified, which files are in staging area and the commit history.
* To add the files to the staging area
  In command prompt → **git add filename**
  (After adding check the status, the file will be in the staging area)
* To remove the file from the staging area
  In command prompt → **git rm –cached filename**
* To add multiple files to the staging area
  In command prompt → **git add .**
  All the files in that location will be added to the staging area.
  If we want to remove any file from staging area, we have to remove each file individually
  Multiple files cannot be unstaged together

**COMMIT**

After adding the files to the staging area we have to commit it
In command prompt → **git commit -m "descriptive message"**

To see the file which is committed
In command prompt → **git log**
The above command will give the details of all commits

**git log --oneline** → This command will give commit ID and commit messages

After commit, if we make any changes in the file, again we should add the file to the staging area and we should commit again.
Each commit will have different commit id.

To undo the commits, there are 3 ways

* checkout commit
* revert commit
* reset commit

**Checkout commit**

git checkout <commit_id>
To check the previous commits, we use this.
This will not modify the code.

**Revert commit**

git revert <commit_id>
git revert creates a new commit that undoes the changes of the specified commit.
It does not remove the original commit from the history.

**Reset commit**

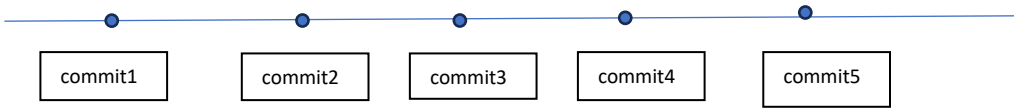
git reset <commit_id>

The above command will permanently delete the data added from the specific commit

Use **reset** when you want to completely erase commits

Eg

Lets assume we are having 5 commits




* By default the HEAD will be at commit 5.
  All the commits will have different commit_id
* Suppose we give git checkout <commit3_id>
  The HEAD will not switch to commit3.
  No changes, no delete. Just The commit 3 will be visible.

* Suppose we give git revert <commit3_id>
  The HEAD will switch to commit3.
  commit 4 and commit5 will no longer be available in Pycharm/VS code.
  It looks like commit4 and 5 is deleted but actually it will be stored in the commit history.
  To revert back the revert operation.
  git revert <revert_id>

* Suppose we give git reset <commit3_id>
  commit4 and commit5 will be discarded permanently.

**BRANCH**

**What is a Branch?**

- By default, Git starts with one branch called **main** (or master).
- Branching allows you to work on new features, fixes, or experiments **without affecting the main code**.

**Why Use Branches?**

- To **develop features separately** from the main code.
- To **experiment safely** (you can delete a branch if it fails).
- To **work in parallel** (each team member can have their own branch).

To create a new branch → git branch <branch_name>
To switch to new branch → git checkout <branch_name>
To create and switch in one step → git checkout -b <branch_name>
To see all the branches → git branch -a
To delete the branch
First switch to master → git checkout master
If the branch is not fully merged
To delete → git branch -D <branch_name>
If the branch is fully merged
To delete → git branch -d <branch_name>
To merge the other branches to the master branch

- git merge <branch_name>

**To Push the code to the github**

To upload the project to the Github repository
Step1 : Login to github
Step2 : Click on repositories, click on **NEW**
Step3 : Give repository name, description is optional
Step4 : Click on create repository.
This will create new remote repository on the github
Step5 : It will give the url, copy the url
Step6 : Go to command prompt, change the path to project(which we wanna push to git)
cd <project_location>
Step7 : check git status
Step8 : Check if the files we want to push is added to the commit area, if not add them.
Step9 : To push → **git push url master**
url → repository url from the remote repo

The code will be reflected in our new repository.

After pushing the code to the github and we make the changes again,

- Add the files to the staging area
  git add <filename>
- Commit
  git commit -m <message>
- git push url master

**To pull the code from repository**

- Open the command prompt
- Give the location where you want to pull the code
  cd <location>
- git clone <repo_url>
  eg : https://github.com/username/repo_name.git
- The code will be downloaded to the Project location and will be visible.
- To update the cloned project
  Open command prompt
  * cd <cloned_repository_location>
  * git pull