***What is GIT?***

* *Git is a version control system /software that works on our local computer that helps to track changes in the code.*
* *It is very popular, free & Open source, fast and scalable tool.*
* *It is useful to track history of our project.*
* *It helps us to collaborate with other stake holders of project as well.*
* ***git --version*** *(Used to check git installed or not and its version)*
* ***pwd***  *(Present working directory)*
* ***cd*** *(Used to change directory/folder)*
* ***cd ..*** *(Used to come out of current directory)*
* ***mkdir <****fileName****>*** *(Used to create new directory with filename specified)*
* ***ls*** *(Displays all the files inside repository)*
* ***ls -a*** *(Displays all the files hidden & unhidden files of repository)*
* ***Git configuration using GitBash***
* ***git config --global user.name “yash05199”***
* ***git config --global user.email”*** [***yashkumbhar05n@gmail.com***](mailto:yashkumbhar05n@gmail.com) ***“***
* ***git config --list*** *(Displays the credentials of user like Username & User mail ID).*
* ***Git Commands:-***
* ***git clone <Repo\_Link>*** *(Cloning/Creating a Copy of repository on our local machine from remote i.e GitHub)*
* ***git status*** *(Displays the status of code , changes made in some folder, inside repository)*

***Types of status displayed after git status :-***

* *untracked 🡪 New files were added that git doesn’t have a track of yet.*
* *Modified 🡪 Newly changes made*
* *Staged 🡪 File is ready to be committed*
* *Unmodified 🡪 File is unchanged/no modifications made.*

*Any new files added or modified or added needs to be added first with*

***git add*** *(filename) or* ***git add .*** *(Adds all files)*

* ***git add*** *(Adds new or changed files in our working directory to the Git staging area)*
* ***git commit*** *(It is the record of change)*

***git commit -m “****Some message****”***

* ***git push*** *(Upload local repo content to remote repo)*

***git push origin main*** *(origin ->it is default repo name from where we took clone, main -> branch name)*

* ***git init*** *(Used to create a new git repo from Terminal)*

***cd .. -> mkdir NewRepo -> git init -> (Git repository Created)***

* ***git remote add origin <repo\_Link>*** *(Add new remote (GitHub Repo) whom we will call as “origin”)*
* ***git remote -v*** *(It will display current GitHub repo of remote)*
* ***git branch*** *(Display the current branch of repo)*
* ***git branch -M main*** *(It will rename the current branch name to “main”)*
* ***git push -u origin main*** *(If we are going to work on same project for longer period of time, then we don’t need to write origin main multiple time only this command is sufficient, we need not to use origin main everytime now, only write* ***git push****)*
* ***git checkout <****branchname****>*** *To navigate from one branch to another branch*
* ***git checkout -b <****new****\_****branchname****>*** *Creates new branch with new****\_****branchname*
* ***git branch -d <****branchname****>*** *Deletes branch specified with branchName*

***(If we want to delete specific branch , we need to make sure we aren’t in that branch so first change the branch and then delete the branch)***

***Way 1:-***

* ***git diff <****branchname****>*** *(Used to compare the other branch with main branch)*

***git diff main*** *(Will compare and display the changes of our current branch with main branch)*

* ***git merge <****branchname****>*** *(This command will merge 2 branches)*

***Way 2:- Create a PR (Pull Request)***

* ***git pull origin main*** *( Used to fetch and download content from a remote repo and immediately update the local repo to match that content )*

***Resolving merge conflicts****:- It is an event that takes place when Git is unable to automatically resolve differences in code between two commits. When 2 branches have commits on same file and on same line then this conflicts arise. Git gets confused as which changes should it accept in repo. Ways to resolve a merge conflict:-*

1. ***Create PR (Pull Request)***
2. ***git merge****Manually change the code and then* ***git merge.***

***Undoing changes:-***

***Case 1:- staged changes (****Changes which are added but not yet committed****) (git add )***

* ***git reset <filename> (****Will reset all add changes made inside a file****)***

***git reset (****Will reset all add changes inside multiple files****)***

***Case 2:- committed changes (Single commit) (git add +git commit)***

* ***git reset HEAD~1 (****Latest commit is stored in Head , this implies git will reset Head one step back ) For eg:- We had 4 commits Last commit is Head(Head~1) means 4th commit will be reset and we will be on 3rd commit****)***
* ***committed changes (Multiple more than one commits)*** ***(git add +git commit\*n)***

***git reset <hash> (Copy the hash of commit which gets generated )***

*-> Moves HEAD + branch to <hash>,* ***keeps your file changes*** *(in working directory)*

*->Reverts all the changes done after that specific commit whos hash we pasted.*

*->* ***Your changes become "unstaged"*** *(still in your working directory)*

***Use When: You want to undo a few commits but keep the file changes to edit or recommit.***

***git reset –hard <hash> { Copy the hash of commit which gets generated }***

***->*** *Moves HEAD + branch to <hash>,* ***deletes all uncommitted changes***

***->*** ***This deletes all changes after that commit — no way to recover unless backed up.***

***🔍 Use When: You want to completely wipe out all changes and go back to a clean state.***

***Fork:-*** *A fork is a new repository that shares code and visibility settings with the original “upstream” repository. Fork is a rough copy.*

*A* ***fork*** *is a* ***copy of someone else's Git repository*** *under* ***your GitHub account****.*

*It lets you* ***experiment freely*** *without affecting the original project.*

*Search a project ->Click on Fork -> Repo Name -> Create Fork.*

*To merge that code with main original code 🡪Create a new Pull Request****.***

***Why do we create branches in repository?***

*🡪Many collaborators can work parallelly on repository so one developer need not to wait for his work to get accomplished even if multiple collaborators are working on same repo.*

***What is GitHub?***

* *GitHub is website that allows developers to store and manage their code using Git.*
* [*https://github.com*](https://github.com)
* ***Project Folders are called as repository (repos) in GitHub.***

***To create a New Repository:-***

*Repositories -> New -> (Give name to repo) -> Access (Public/Private) ->Add a Readme File ->->Create a repository.(Done)*

***Initial Commit- >*** *The very first changes made to repository is called initial commit.*