# Git Commands

- ls - listing all the folders present in that directory / Folder
- mkdir project - create a folder name project
- cd project - change directory → Manually → double click on project folder
- ls -a - it shows all hidden files
- git init - Initialize a empty git repository in the folder
  ↳ Any file that start with . (dot) are hidden in Linux / Mac OS
- ls .git - it gives what is inside git folder
- touch names.txt - Create new file in Linux
- git status - it give status that something is added/ deleted or modified in the directory it shows untracked files as well as tracked one
  ↳ Untracked files (History of directory → project is not maintained)
  ↳ If we change anything in our directory i.e. project how would ~~any~~one know that there is something change in project. So to solve this we use-
- git add names.txt - it add untracked files to git repository so that history of directory → project is maintained. (Select that file and moved it to staging area, marking it for inclusion in the next commit)
- git commit -m "names.txt file added" - A shortcut command that immediately creates a commit with a passed commit message. So that they are permanently store in git history. (m → message)
- git config --global user.email "youexample@r.com" - You can use the git config command to change the email address you associate with you git ~~commands~~. commits.
- git config --global user.name "Name" - To set your name (Git username)
- vi names.txt - To go inside text file → ~~To~~ use Press i → To write in the File
  → Press Esc → Press : with shift → Press any key → Enter
- Cat names.txt → To see what is inside the file
- git restore --staged names.txt - If history is already saved again you are doing same thing then you can restore it from saving again and again. (→ to discard changes in working directory)

- git log - To see entire history of project all the commits that were made in the history

rm -rf names.txt - To delete names.txt file

→ If we deleted or modified file by mistake then then we use git command(s)

git reset (hashcode that we copied) - It remove all the commits which are above the commit whose hashcode we copied.

→ The commit we want to remove → we copied the hashcode just below it

- git add. - adds a change in the working directory to the staging area.

→ Staging area is files that are going to be a part of the next commit, which lets git know what changes in the files are going to occur for next commit.

↳ Staging area also referred to as untracked files.

- git stash - Git stash is a built-in command control tool in Git locally stores all the most recent changes in workspace and resets the state of the workspace to the prior commit state. (You can hide your changes with this command)
(Saved working directory and index state WIP on master)

↳ Master is the default branch in Git

- git commit - It captures a snapshot of the project's currently staged changes. It capture the state of the project at that point in time.

- git stash pop - It helps us to remove or throw away the latest or the topmost stash

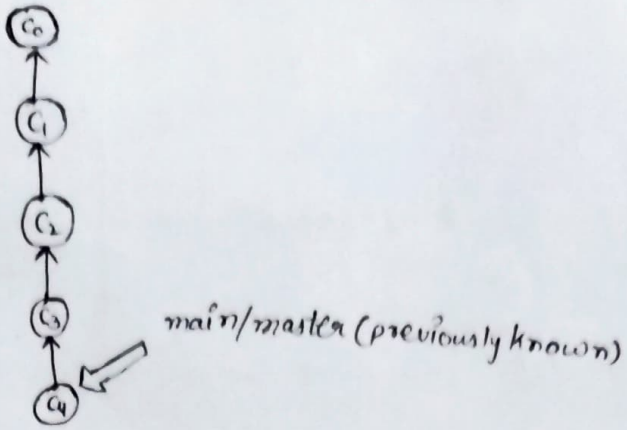
- git stash clear - Deleted the most recently created Git stash.

- git remote add origin URL - Obtain the remote repository that a project was originally cloned from.
  - remote: working with the URLs
  - add: adding a URL
  - origin: name of URL
  - URL: what is the name of URL you are going to add

- git remote -v - You can verify that remote URL has changed with this command.
  ↳ it give all the URLs that are attached with a folder.

- git push origin master - To share changes in the URL
  - origin: URL
  - master: branch

- **branch** - When we make commit it makes ~~bra~~ internally it make branch like structure. Git branches are effectively a pointer to a snapshot of your changes.




↳ Don't make commit directly on main branch because the code use by people can't affect ~~not~~ create another branch and make commit in it.

↳ Because our code that is not finalized yet might contain some errors, so all code on which you are working go on separate branch so that users are not affected

- git branch **feature** - The idea behind it is that all feature development should take place in dedicated branch instead of main branch.
- git checkout feature - It lets you ~~imagine~~ navigate between branches created by git branch.
- HEAD - HEAD can be termed as a special ref that points to the current commit.
- git checkout master - Command it is used to get back to master.
- git branch -D <branchname> - Delete the branch
- git switch - - Command used to switch between master and branch
- git switch - <name> - Command used switch to particular which is written
- Ctrl + C - To stop cloning
- git clone URL - It copies an existing Git repository → it primarily used to point to an existing repo and make a clone or copy of that repo in, a new directory at another location.

↳ From where we forked that URL is called upstream by convention

↳ When you create your own copy and create changes in your copy how do you make sure that whatever you change is visible in the main project You request it via Pull requests then people will review your code suggest some changes and you will make those changes and when it merged your code sample change and visible in the main branch.

↳ For every new ~~pull~~ feature you have to pull request but in a branch you can pull request only once for many pull requests you have create more branch. If a branch already have pull request associate with it it will not allow you create new pull request.

↳ Never commit on main branch for eg. if you are coorking on 10 features every changes you make it is doing on one pull request how difficult it is to manage.

- git fetch --all --prune - fetch all the commits
  ↓ which are delete that also
- git reset --hard upstream/main - Reset the main branch of by origin to the main branch of upstream

↳ This can also be done with button feteh upstream

- git pull upstream main → Same as the fetch command (above)

↳ If we coant to merge all the commits you have made into one commit

- git rebase -i <hash code> - It squash all the commits into one
  ↑ Iteractive environment

↳ Merge Conflicts - when two person change the same line of code