==================================================================================================================

First write:

Pwd == (Print Working Directory) == It shows what git is looking at

\_\_Try this if u forget( commands are step by step , i dont think you can jump unless if write the full command)

First write , "cd~" , then write , "cd desktop" , then write , "cd (any folder) ,

> IF you write cd (any folder) directly , you will get error

> IF you write in full like , "cd ~/desktop/(any folder)/" , then you will go directly to the folder u wished

=========================================================

=========================================================

If you want help in a particular command line , write "help" &enter , you will be show help menus .

Chose the menu by writing "git help (menuExample) , browser will be launched about that menu in detail

=========================================================

=========================================================

How to commit as far as i know

→ First you make a folder/directory , you write ( Git init == Starts Git project )

→ When you add/remove/delete a file in a folder/directory , you write ( git add . == add all of the changes made to our project )

→ OR if your adding a particular file , write ( git add (writing the exact folder) == Exact folder will be added to changes )

→ When done , you then write ( git commit -m "(Your Comment/Message)" ) == Saves your changes/Takes a snap shot/when ever you do commit files ,it adds in the repository

→ Then write ( git status ) to check if changes have been made

→ Extra ( git log == Shows MY commit history )

- If it shows "On branch master nothing to commit, working tree clean" , then everything is stored

- You will get fatal errors ALOT, either repeat steps or spam the nut out of git, especially writing git status before executing command.

- "Untracked files" are not commited files(not keeping in track of them)

=========================================================  
=========================================================

How to connect your git terminal/bash(your local repository) to your public repository/server on github  
  
- First initialize the folder with “git init  
- Then , make new repository on git hub  
- Copy URL   
- Then type ( **git remote add (public repository NickName) (paste URL from git)** ) Connects your local pc to online server  
- Finally ( **git push -u (public repository NickName) master**  ) in order to push your files in your public GB acct  
- Done  
  
  
=========================================================  
=========================================================

-------------------------------------------------------------------------------------------------------

> cd == Changes directory/folder

> ~ == Home

> cd ~ == Changes folder/directory to home

> cd .. == Backwards/Go back one folder/directory back (I do not think there is a command to move forward)

> cd ~/desktop/ == Changes your working directory to desktop (i think)

-------------------------------------------------------------------------------------------------------

> mkdir == Makes new folder i think (or makes new directory) ((but it does not work)

> pwd == (Print Working Directory) == It shows what git is looking at

> Ls == list stuff in that folder/directory

> Ls -La == Shows you hidden files in an empty folder(despite folder being empty) (Dont Touch Those Hidden files)

> clear == clears your git screen

-------------------------------------------------------------------------------------------------------

> git == list random stuff that idk but it shows the git is installed

> **git init** == Starts Git project

> git --version == Shows you latest version

> git config --list == Shows you settings (idk whats the use of it )

> git help == Opens help section

> git config --global user.email == "you@example.com"

> git config --global user.name == "Your Name"

> git config --list == list user name and email

> git config user.email == Pops your email

> git add . == add ALL of the changes made to our project

-- the small dot (.) means folder or directory

> **git add (write the exact folder)** == Exact folder will be added to changes

---The "add" means files are ready to be commited(also know as staging area)

> **git commit -m "(Your Message)"** == Saves your changes/Takes a snap shot

---when ever you do commit files , it adds in the repository(from the working directory)

> git log == Shows my commit history

> git log --author="(Username)" == Shows commit history of that particular User (even yourself)

-- MAKE SURE your that you dont leave capital letters aswell

> git status == Show current on commit of a particular folder/directory

-- Green text a in staging area(ready to be commited/going in repository)

-- Red text is "working copy" or "Untracked files" (Its only on our pC, not in staging or repository) or (not keeping in track of them)

> **git diff** == Shows you modified changes . (Only compares files from Working area to repository || STAGING AREA IS NOT COMPARED)

-Red is original , Green is new changes

> *git diff --staged* == Shows you modified changes , (ONLY compares files from staging area to repository)

> **git rm (name any file)** == Removes files  
-- Removes files from repository & working area but its still in staging area   
-- Commiting it to remove from staging area

> git mv ( txt file ) ( txt File) == Changes Name  
> git mv ( txt file ) ( Folder)/(changes name of txt when writing some here)   
> **git commit -am “any comment”** == Commits EVERY FILE directly into repository area from working area without adding files in staging area  
-- (skips staging area)  
> git checkout -- (Exact file Name) == **Restores your deleted file/modified changes**  
> **git reset head (Exact file name)** == Removes file from staging area to Working area  
-To prevent it from commiting to changes  
> **git checkout (commit id number written in gold)** == -Goes back to that particular repository/commit ID FROM the current repository  
> **git checkout master**  == goes back to the master (i think , dont know much about this)  
> *git checkout (commit id no.) -- (file/.txt name)*  == brings you the copy of the file/.txt from that old repository your written (commit ID no.) into your main repository  
> **git remote add (public repository NickName) (paste URL from git)** == Connects your local pc to online server  
> **git push -u (public repository NickName) master**  == Pushes all repositories from local to main github server   
> *git remote* == shows public repository NickName  
> git remote -v == Shows the github URL attached to this folder  
> git stash == saves staging area somewhere   
> git stash pop == brings back staging area to working directory  
> git stash clean == Deletes the staging area save  
  
  
  
  
   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==   
> ==