Git CMD lines , learning again

How to commit a folder/repository

→ First you make a folder/directory either how normally you would or **mkdir (then write something here)** , then you write **Git init**

== initializes a (git) project/repository in the folder/file that we are currently at. Or it keeps track of changes in that file/folder

→ 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 (Then write 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 , write **git log**

== Shows MY commit history

==========

Commits I use

Remember , the double quotation marks (“) changes in notion , which means copy pasting notes from notion to git (or any other place maybe) actually changes the code entirely cuz the comma changes to a retarded fish

> **pwd**

= print working directory / or it shows where we are

> Git --version

= gets latest version

> git config --global user.name "Umer Khan G"

= sets your username (i think only on git and not github)

> git config --global user.email "ur email"

= Email is set on git , so any code changes you make here(on git) , you can then upload them on github

> git config --list

= lists ALL of your settings like user email/name

> git config user.email

= list email

> git config user.name

= list user name

> pwd

= It shows what git is look at right now/ shows where git is at

> Ls

= list everything what git is looking at/ where it is

> ls -la

= lists everything **including hidden** files (dont touch these files)

> cd ~

= changes directory to home (what ever the pc considers as home)

> cd ..

= changes directory by going one step back

> mkdir (then name it , else it wont work)

= makes a folder for you

> **git init**

= initializes a (git) project/repository in the folder/file that we are currently at. Or it keeps track of changes in that file/folder

> **git add .**

= commits/add **ALL** the changes we did to our Project/folder (like creating a file)

- the small dot (”.”) means **all**

> **git add “name of the file”**

= commits/add that particular file mentioned/named

- git add “name of the particular file to be more specific if file/folder has spaces
  ”(difference is you added commas after the git add)

> **git commit -m “any message ur wrote between the commas”**

= It commits your project to your repository

- ***Files are being tracked***
- its actually takes a snapshots if we mess up in a code somewhere between our project and which helps us where do we want to go back to

> **git log**

= Views commit history

- (history of git commit -m “any message ur wrote between the commas”)

> **git status**

= Shows updated status on particular ***folder/project***

- Green text a in staging area but not yet committed or ready to be committed files
- Red text is "working copy" or "Untracked files" (Its only on our PC, not in staging or repository) or (**Git is not keeping in track of the files**)

> git diff

= Shows ***texts*** that has been modified

- Green shows added text
- Red shows removed text
- compares working copy /untracked files to repository

> git diff --staged

= Same as “git diff” except it compares the staging area to repository (staging area is ready to be committed files)

> git rm (name of the file)

= removes files

- We still need to commit the files
- use “” to name the file more specifically

> git mv (txt doc no.1) (txt doc no.2)

= it kinda moves the file but Its basically renaming a file

> git mv (name.txt) (name folder)/(renames .txt)

= moves the text file to the the folder and /renames it

> git commit -am “comment here for notes”

= ADDS ALL OF YOUR FILES TO YOUR REPO DIRECTLY

- skips **untracked files(red)** and **staging area(green)** but commits your project/folders directly to your repository
- No git add .
- No git commit -m
- It some times doesnt work on untracked files for some reason

> git checkout -- “file name”

= undo the local changes or the untracked files(red) and returns the changes to the original committed files

> git reset HEAD “file name”

= Undo the **staging area files(green)** and returns the changes to untracked files(red)

> git checkout -- (the numbers when you write git log) (file name)

= Gets the Old Versions from the Repository

- You dont have to write all the numbers
- ***I COULDNT TEST THIS ONE OUT BECAUSE I COULDNT FIND ANY FILLED FILES AS EXAMPLES***

>

=

>

=

>

=

>

=

>

=

>

=

>

=

|>
||

|>
||

|>
||

|>
||

|>
||

>

=

==========:

Commits I might not use

> git help (name of the topic , like commit) = Opens a whole web tab/page on that particular topic

> =

>=

>=

>=

>=

>=

>=

>=

|>

||

|>

||

|>

||

|>

||

|>

||

|>

=