# GIT

\*There are some comnads that will work on the windows terminals and others that will work on the the ubuntu/gitbash shell

Some commands and things

Cd -> change directory

Cd / is when you go to the top level directory (absolute path )

Clear -> removes things in the terminal

Cd (Specific folder) \*\*\* will move to the specific folder

cd.. this will go to 1 folder above the current folder

Mkdir -> Creates a folders and makes that the current directory

Mkdir -> (Specific folder) \*\*\* will create a folder/file in that specific file path

Ls -> lists all of he files in the specific directory

For windows user we will have to install a linux shell to access the functionality unlike in mac

~ means the home directory

Cd~ go to home directory

-f means force \*to foreculyy do any activity

Press tab for auto-completion when you partially type something

There is no control v and p in the linus terminal so you have to right click to paste.

Mnt -> is the mount directory as in the directory where you are running the operating system (C)

To access paths in this specific directory then you must do the following:

Cd ***\mnt***\c\users\username

Rm -> remove file \*\*-rf is for removal of folders

To remove folders recursively

Rm -rf (folder )

Touch (filename) -> Creating a file

Creating files I n power shell or cmd

Echo $nill >> (filename)

GIT CHEAT SHEET  
<https://github.com/joshnh/Git-Commands>

### Getting & Creating Projects

| **Command** | **Description** |
| --- | --- |
| git init | Initialize a local Git repository |
| git clone ssh://git@github.com/[username]/[repository-name].git | Create a local copy of a remote repository |

### Basic Snapshotting

| **Command** | **Description** |
| --- | --- |
| git status | Check status |
| git add [file-name.txt] | Add a file to the staging area |
| git add -A | Add all new and changed files to the staging area |
| git commit -m "[commit message]" | Commit changes |
| git rm -r [file-name.txt] | Remove a file (or folder) |

### Branching & Merging

| **Command** | **Description** |
| --- | --- |
| git branch | List branches (the asterisk denotes the current branch) |
| git branch -a | List all branches (local and remote) |
| git branch [branch name] | Create a new branch |
| git branch -d [branch name] | Delete a branch |
| git push origin --delete [branch name] | Delete a remote branch |
| git checkout -b [branch name] | Create a new branch and switch to it |
| git checkout -b [branch name] origin/[branch name] | Clone a remote branch and switch to it |
| git branch -m [old branch name] [new branch name] | Rename a local branch |
| git checkout [branch name] | Switch to a branch |
| git checkout - | Switch to the branch last checked out |
| git checkout -- [file-name.txt] | Discard changes to a file |
| git merge [branch name] | Merge a branch into the active branch |
| git merge [source branch] [target branch] | Merge a branch into a target branch |
| git stash | Stash changes in a dirty working directory |
| git stash clear | Remove all stashed entries |

### Sharing & Updating Projects

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Command** | **Description** | | --- | --- | | git push origin [branch name] | Push a branch to your remote repository | | git push -u origin [branch name] | Push changes to remote repository (and remember the branch) | | git push | Push changes to remote repository (remembered branch) | | git push origin --delete [branch name] | Delete a remote branch | | git pull | Update local repository to the newest commit | | git pull origin [branch name] | Pull changes from remote repository | | git remote add origin ssh://git@github.com/[username]/[repository-name].git | Add a remote repository | | git remote set-url origin ssh://git@github.com/[username]/[repository-name].git | Set a repository's origin branch to SSH |   **ommand** | **Description** |
| git push origin [branch name] | Push a branch to your remote repository |
| git push -u origin [branch name] | Push changes to remote repository (and remember the branch) |
| git push | Push changes to remote repository (remembered branch) |
| git push origin --delete [branch name] | Delete a remote branch |
| git pull | Update local repository to the newest commit |
| git pull origin [branch name] | Pull changes from remote repository |
| git remote add origin ssh://git@github.com/[username]/[repository-name].git | Add a remote repository |
| git remote set-url origin ssh://git@github.com/[username]/[repository-name].git | Set a repository's origin branch to SSH |

### Inspection & Comparison

| **Command** | **Description** |
| --- | --- |
| git log | View changes |
| git log --summary | View changes (detailed) |
| git log --oneline | View changes (briefly) |
| git diff [source branch] [target branch] | Preview changes before merging |

Remote repository -> Online repo

Repo -> Full timeline of the code

Genera process

Working directory -> Git Add -> Staging area -> Git Commit -> Local Repository> Git push -> Remote repository

Creating new repository using local folers->

Initialize the repository

* Git init

Checking staging area status ->

* Git status
* Git add (filename)

Adding all files

* Git add .

Removing and deleting files -> when done manually you still need to do the following to add the files to the staging area

Deleting files from stagin area fully

1. Git rm (filename) -f

Deleting file temporarily

1. Git rm –cached (filename)

Committing

* Log on before you can commit

Loging in

* Git config –global user.email “emal address”
* Git config –global user.name “username”

Finding help for git

* Git --help
* Git commit –help

\*\* Committing tracked files

* Git commit -m”message”

Committing untracked files

* Git commit -a m”message”

Tracking all of the commits

* Git log

Moving between commits

\*\*The commit that has the {head -> master} is the current version and for you to access previous versions of the code on your local you must use the specific commit ID

The following must be done

* Git checkout (commit ID)

To go back to the latest code you then need to go back to the master using the following

* Git checkout master

Aftervwhcih you can do git log and find all other existing logs to choose from

Since there are no branches we only go up and down but if there are we would also need to explore the branches of code

* Reverting and resetting-> Making the changes permanently

Revert is safer than reset. -> Revert is for only one commit which is as follows:

* Git rivert (commit ID)
* It does not delete the other commits

Reverting the revert requires one to go back to the master

* Git master checkout

After then we must revert the revert

* Git revert (commit ID with revert)
* Git reset (commit ID) => Either does hard reset or soft reset but it is not recommended
* Git reset –hard commit ID -🡪 Removes commint
* --soft does not remove the commit/files
* --mixed are files that are in working directory that just need to be committed but have not yet been deleted -\*\*\*\*\* This is is the default of the git reset

If there are files you do not want to track then you use git ignore->Autogenerated files

You must create a file that has the extension .gitignore and you must add the names of all the fikes that need to be ignored

* Touch a.gitignore

Then open the file and names of all files

After this you must remove all of the cached files that are track so they can be ignore by the git ignore using the following:

* Git rm -r –cached

If you want all the files in a specific folder to be ignored then in the gitignire file you would put the name as follows

* Foldername/\*

If gitignore is added before files are committed then there iss no need to clear or remove the cach so that the git ignore can work

Branches-> separate development path

* One for bug fixes
* One for release to the production
* One for development

At end you can merge the branch to the master

--\*\*\*you must make an initial commit before branching or the branch and the master will be seen as the same

Creating a branch is as follows:

* Git branch (branch name)

Alternatively you can use the following where you change to the existing branch you have created

* Git checkout -b branchname

Listing branches:

* Git branches -a \*\* a is for all

Deleting branches:

* Git branch -d branch\_name

Mergin branches:

* Git merge name\_of\_branch

Github

Fork -> you take a copy of the repo and add it from your account

Create remote repo amd connecting to git hub

Creaete online on git hub with no files or readmes etc

Go to desktop to specific directory on ubuntu

Git init

Connect to github using terminal by adding origin and the ssh link

Git remote add origin https/ssh..

Use https as it is faster less configuration

Check if it works or is connected

Git remote -v

Pull requests

Git pull origin master ->

Git pull (branch name)

Pushing or merging branches from current branch

Git push -u origin master

Pushing another branch to the master brach

Git psuh origin branch name

Deleting branch on git huub

Git push origin –delete (branchname)

SOURCE TREE

Open the thing then click on local then click on add repository then make sure you do not have create repository on account active

First create a file in the folder before the menu will change