Git Cheat Sheet By Mayur Purushvani

-> What is Git :

Git is a version-control system

It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

One big advantage is that if you lost your files, you can easily recover.

-> Version Control Tool (VCS)

-> 3 Types :

-> Revision Control System (RCS)
-> Centralized Version Control Systems (CVCS)
-> Distributed Version Control Systems (DVCS)

Untracked - Add new File on Git (Automatically goes in staged area)
Staged - one type of Ground. When We commit the file itwill goes in Unmodified section.
Unmodified - 2 options - 1. Edit the file and modified it. 2. Remove the file and it will go to the Untracked section.
modified - modified file Automatically goes in to the staged section.

-> COMMANDS :

-> Git Basic :

Go to the loction (Open Gir-Bash)

git init

ls -lart (list the git)

git status (Shows the status of files - Add a new file (Goes in UNTRACKED AREA))

git add index.html (add the file on git - Goes in STAGED AREA)

git commit (When you commit the File, It will open new terminal, Then Press the 'i' key to enter the message. and type ':wq' to exit from terminal)

OR

git commit -m "[MESSAGE]"

touch aboutus.html (To Create an empty file)

git add -A (Add all the files you have created on git in once - To add the files on staging area)

git commit -a -m "[MESSAGE]" (It will commit all the files with message (Skip the Staging area))

git rm [FILE_NAME] (It will remove the file from staging area and also from harddisk like DELETE the file)

git rm --cached [FILE_NAME] (It will remove the file from staging area only... and it will goes in Untracked files)

git status -s (It will shows the changed files only.. While it is from staging area (In Green color) or it is from local path(in red color))

->HOW TO RECOVER THE OLD CODE :

git checkout contactus.html (For SINGLE FILE)

git checkout -f (For multiple Files)

-> Git Branches

git branch [NAME_OF_BRANCH] (Create a new branch. It will not modify in master branch)

git branch (List all the branches avail in your project)

git checkout [NAME_OF_BRANCH] (For Switch the branch)
-> You can also changed the file in feature section and add it on the brach.

git branch -d [BRANCH_NAME] (To delete the branch)

git branch --move [EXISTING_BRANCH_NAME] [NEW_NAME_OF_BRANCH] (Rename the branch) (This is only for local)

git push --set-upstream origin [NEW_NAME_OF_BRANCH]

git push origin --delete [BRANCH_NAME] (To delete the branch at remote server.)

git branch -v (To see the last commit of each branch)

git branch --merged (To see which branch are merged)

git branch --no-merged (To see which brandh are nor merged)

git branch --all (To see all the remote based branched and local branches)

-> Git Clone

git clone [URL] [FOLDER_NAME] (For clone any project to your computer.)

-> Git Merge

git merge [BRANCH_NAME] (If you have a branch named FEATURE, and you have to merge it into the master branch,

Then, use merge command. First switch into the master branch using "git checkout master" and then type merge command.)

git checkout -b [BRANCH_NAME] (Create a new branch and Automatically switched into that branch)

git show (Show the logs of files)

git show [PARTICULAR ID] (use the git log command and copy the ID)

-> Git Rebase

git rebase [BRANCH_NAME]

git pull --rebase origin/[BRANCH_NAME]

git rebase --continue

git remote show (Show the name of the aliase)

git ls-remote (Show the proper remote details with all branches)

git rebase [BRANCH_NAME] (First chane gomething in any branch, then ass and commit it. then try to rebase it. It is simply rebase the branches and copy the code from one branch to another.)

TO COPY THE CODE FROM ONE BRANCH TO ANOTHER

-> git fetch origin

-> git merge --no-ff origin/[BRANCH_NAME]

-> Git push

git push -u origin FEATURE
(Insert the new branch on remote server)

->Git Config

git config --global user.name "[Your Name]"

git config --global user.email "[youremail@yourdomain.com]"

-> Git Log

git log (Shows the last commits)

git log -p -5 (Shows the particular last records)

-> Git Diff

git diff (If you edit something in the file, then it will show the last change og file and current change of file)

git diff --staged (It will show the difference between last modified and current file content, (Only when we add the files on git))

-> GIT REMOTE CONTROL

-> git remote add origin https://github.com/mayurpurushvani/MayurTraders.git

-> origin is the url name.

-> git remote (Show the total remotes you have created)

-> git remote -v (Shows the full url from where you have fetched and push the code)

-> Before push the files on remote server, You have to add the SSH certificate.

-> For SSH certificate, Follow below steps....!

(This all steps is shown in the github website :

https://docs.github.com/en/free-pro-team@latest/github/authenticating-to github/connecting-to-github-with-ssh)

--> ssh-keygen -t ed25519 -C "mayurpurushvani@gmail.com"

--> eval $(ssh-agent -s)

--> ssh-add ~/.ssh/id_ed25519

--> cat ~/.ssh/id_ed25519.pub

When you write above command, it will generate the SHA key, and copy that SHA key and paste it on you git hub account's SHA key.
(Go to the settings -> SHA key -> Add SHA key)

-> git push -u origin master (From origin to master - push all the files online using this command.)

-> .gitIgnore file

-> CREATE a .gitignore file and write "mylog.log" in the file.

-> Then, create a new folder named logs.

-> Inside the logs flder create the file named 'mylogs.log' OR you have to write "*.log" OR "*.cpp"

-> then, check the git status

-> It will not display the mylog.log file for commit. It ignores that file. If any file you don't need to commit then simply put the file name on the ".gitignore" file.

DIFFERENCE BETWEEN MERGE AND REBASE :

-> MERGE :

This creates a new “merge commit” in the feature branch that ties together the histories of both branches

-> REBASE :

- git rebase is to never use it on public branches.

- For example, think about what would happen if you rebased master onto your feature branch

- The rebase moves all of the commits in master onto the tip of feature.

- The problem is that this only happened in your repository.

- All of the other developers are still working with the original master.

-> Interactive Rebasing :

-Interactive rebasing gives you the opportunity to alter commits as they are moved to the new branch.

-This is even more powerful than an automated rebase, since it offers complete control over the branch's commit history.

-Typically, this is used to clean up a messy history before merging a feature branch into master.

-> GITLAB :

-> GitLab started as an open source project to help teams collaborate on software development.
-> GitLab's mission is to provide a place where everyone can contribute.

-> git remote add gitlab [URL]

-> git clone [URL]

->FORK :

➔ Fork is used for sharing the code between two different accounts repositoried.

➔ We can use form by two things :

1. If someone added public repository and you need that repository to your account, Then simply you can fork that repository to your account, and you get all that files in your repository.

2. If someone have a trouble in some code, Then We can fork that repository and we gave the solution of that file. And generate a pull request. Then, the

actual user can approve that pull request and we gen the solution. So, This two ways we use the fork functionality.

REFERENCE :

https://www.youtube.com/watch?v=gwWKnnCMQ5c
https://git-scm.com/book/en/v2/
https://github.com/mayurpurushvani/
https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/connecting-to-github-with-ssh
https://git.wiki.kernel.org/index.php/GitHosting
https://git-scm.com/book/en/v2