GIT CHEATSHEET

COMMAND	EXPLANATION(USES)
`git config`	Set configuration Variables
`git config --global user.name "Your name"`	Set your name
`git config --global user.email "Your email"`	Set your email address
`git config --global core.editor "Your IDE"`	Set your default text editor
`git config --list`	To list all the settings
`git help config`	To get help while using git
`git init`	To track project in git
`git add`	To track a new file in a project
`git commit` or `git commit -m 'Commit message'`	To commit after staging a file
`git commit -a -m 'Your message'`	Helps to commit while skipping staging a file(Directly add & Commit)
`git commit -amend`	It overwrites your previous commit
`git clone[url]`	To get a copy of repository
`git clone /../../` or `git clone file://`	To clone a local repository
`git status`	To check which files are in which state
`git status -s` or `git status -short`	Give simplified output from this command
`.gitignore`	Files that you don't want Git to automatically add or even show you as being untracked
`git diff`	To see the changes in your files
`git diff --staged`	To see what you have staged and want to commit

COMMAND	EXPLANATION(USES)
`git diff --cached`	It shows all the difference between staged files
`git rm`	To remove a file from git
`git rm --cached`	To keep file in working tree but remove it from your staging area
`git mv file_from file_to`	Helps to rename a file
`git log` or `git log -p -2`	To see the history of commits
`git log --stat`	To see some abbreviated stats for each commit
`git log --pretty=online`	Helps to prints each commit on a single line
`git log --pretty=format:"%h - %an ,%ar :%s"`	It allow you to specify your own log output format
`git log --pretty=format:"%h %s" --graph`	It adds a nice little ASCII graph showing branch & merge history
`git log --since`	This helps us to check the history of weeks , months
`git log --Sfunction_name`	Shows a commits that introduced a change to the code that added or remove that string
`git log --online --decorate --graph --all`	
`git reset HEAD <file>`	Helps to unstage the commit
`git checkout -- <file>`	Helps to discard or revert the changes
`git remote add <shortname> <url>`	To add a new remote repository as a shortname
`git remote -v`	It shows the URLs that git has stored for the shortname to be used when reading & writing
`git fetch [remote-name]`	To get data from your remote Projects
`git pull [remote-name]`	To get data from your remote Projects
`git push [remote-name][branch-name]`	This works only if cloned from a server to which you have write access & if nobody has pushed in the meantime
`git remote show [remote-name]`	To see more information about a particular remote
`git remote rename file_from file_to`	To change the remote’s shortname
`git remote rm <file>`	To remove a remote file
`git tag`	It helps us to tag specific points in history
`git show`	To see the data along with the commit that was tagged
`git push origin [tagname]`	Helps to transfer tags to remote server

COMMAND	EXPLANATION(USES)
`git push origin --tags`	To push all tags to remote server at once
`git config --global alias.co checkout` or `git config --global alias.br branch` or `git config --global alias.ci commit`	It helps us to set up git commands for easily use
`git branch <branch name>`	It creates a new branch
`git checkout <branch name>`	Helps to switch to an existing branch
`git merge <branch name>`	It helps to merge the branch to the master(main) branch
`git branch -d <branch name>`	Helps to delete the branch when no longer needed
`git branch -D <branch name>`	It helps us to force delete the branch when not done with -d
`git mergetool`	Helps to use the graphical tool to resolve the issue
`git branch -v`	Shows a simple listing of your current branches
`git branch --merged`	To see which branches are merged into the branch you're currently on
`git branch --no-merged`	TO see all the branches that contain work that haven't yet merged
`git fetch (remote)(branch)`	It helps to fetch the data from the remote to your local branch
`git push (remote)(branch)`	It helps to push code to remote branch or on server
`git checkout -b [branch] [remotename]/[branch]`	To get your own branch from remote branch to your local branch
`git branch -vv`	This lists your local branches with more info., including the branch is tracking or your local branch is a head, behind or both
`git push [remote] --delete [branchname]`	It is use to delete a remote branch
`git rebase <branch name>`	It takes all the changes that were committed on one branch & replay them on another one
`git rebase --onto<branch 1><branch 2> <branch 3>`	Checkout the branch 1 , figure out the patches from the common ancestor of the branch 2 & 3 and replay them on branch 1
`git rebase [basebranch][topicbranch]`	Rebase the topic branch on top of the base branch without having to check it
`git push --force`	It is use to overwrite the history on the server when we are using command rebase and collaborating with others

COMMAND	EXPLANATION(USES)
`git pull --rebase`	It is used to fetch and merge changes from a remote repository while also rebasing any local commits on top of the updated remote branch.
`git config --global pull.rebase true`	When you are using git pull & want to make a default rebase
`git remote add local_/../../..git`	To add a local repo to an existing git project
`git clone --bare`	It is used to create a bare repository, which is a special type of Git repository that does not contain a working directory.
`git daemon`	It is used to start a lightweight Git server that allows clients to fetch and push changes to a Git repository over the network.