git init (To create local repository)

git remote add origin <https://github.com/96ankur/Hh.git> ( To synchronize both local and remote repository)

git pull origin master

git status

git add “<file\_name OR .(dot) >”

git reset ( this command is used to undo a commit to staged snapshot)

git commit -m "first commit" (-a flag can be used and this will also perform adding to the index. But is won’t work if the file is not already present in the index)

git commit --amend (Passing this option will modify the last commit. Instead of creating a new commit, staged changes will be added to the previous commit.)

git push -u origin master ( -u upstream, for the first time then only **“git push”**)

git checkout <name> (switching from one branch to another branch)

git merge <branch\_name> (whenever you are merging always check to be in destination(master) branch and branch is still available one we have merge it with the destination branch)

git push origin <branch\_name> (this will push changes to a certain branch)

git rebase <name> (result is same as merge but it is used to reduce the number of branches).

git fetch (fetch all the files from the remote repository and store in a separate branch ,that is, it does not merge the changes in the master branch of the local repository, so if we perform git fetch then we have to use git merge separately whereas “git pull =git fetch + git merge” )

BRANCH

git branch <branch\_name> (this will create a branch from the current branch and it will contain all the files that are present in the master branch)

git branch -r (list all the remote branches)

git branch -a (list all the local and remote branches)

git branch (list all the local branches)

git branch -m <branch> (Rename the current branch to <branch>)

git checkout -b <new-branch> (Create a new branch, named as “new-branch”)

DIFF

git diff <commit id\_1> <commit id\_2> difference between two commit

git diff <branch1> <branch2> difference between two branches

OR git diff <branch1>..<branch2>

git diff <branch name\_1> <branch name\_2> ./<filename> difference between files from two branches

LOG

git log

git log --prety=oneline

git log -n <limit> (gives a particular number of commits)

git log --stat (Along with the ordinary git log information, include which files were altered and the relative number of lines that were added or deleted from each of them.)

git log --graph --decorate

STASH

=> No stash after commit only before commit

git stash

git stash pop (Popping your stash removes the changes from your stash and reapplies them to your working copy)

git stash list (get a list of stashes)

git stash apply stash@{1}

git stash save "stash message"

git stash branch <branch\_name> stash@{0} (create a branch from the stash)

git stash drop stash@{1} (remove a particular stash)

git stash clear

CLEAN

git clean -n (show you which untracked files are going to be removed without actually removing them.)

git clean -f (or --force) (remove the untracked files)

git clean -dn (for directory)

git clean -fn (also for directory)

DELETE BRANCH

git branch -d <branch\_name> ( -d is alias for delete)

git branch -D <branch\_name> ( -D is alias for --delete --force)

REVERT

The git revert command can be considered an 'undo' type command, however, it is not a traditional undo operation. Instead of removing the commit from the project history, it figures out how to invert the changes introduced by the commit and appends a new commit with the resulting inverse content. This prevents Git from losing history, which is important for the integrity of your revision history and for reliable collaboration.

git revert <commit\_id>

**Resetting vs. reverting**

It's important to understand that git revert undoes a single commit—it does not "revert" back to the previous state of a project by

removing all subsequent commits. In Git, this is actually called a reset, not a revert.

RESET

--soft from committed to staging area

--mixed from committed to untracked

--hard delete

**git reset –soft <commit\_id>**

=> to revert the changes

first take the first eight digits of the commit hash from “git log” and then

git checkout 8A5BD2FE <file\_name\_to\_be\_reverted>

ssh-keygen (in order to generate a key)

ssh -T git@github.com (for authentication of the key)

git config --global credential.helper store (to store password of git hub)



