GIT Commands with explanation

* Git init

Create new git repo on local

* Git clone <link of remote repo> [folder]

Clone existing git repo on local

Here folder is an optional path to the local folder (*which will be a local repository*). We could have used git-test but we can avoid it. If folder is not given, then Git will create new folder with the name same as the remote repository name. This will also initialize the .git directory inside it and set origin to repo-url. This will also pull code from remote repository into the local repository.

* Git remote add origin <link of remote repo>

links the remote repository to local one using reference name as origin

reference name can be anything

* git status

To know the branch you are on and status of files in working directory and index for that branch

* git add <file/folder>

To add that files or folders to the staging area/index

* git add .

To add all files and folders to the staging area/index

* git add -A stages **all changes**
* git add . stages new files and modifications, **without deletions**
* git add -u stages modifications and deletions, **without new files**

If you added some files in the staging area by mistake, then you can use the following command to unstage them.

* git reset <filename>
* git commit -m ""

To commit the file and move it from staging are to local repo

Message is compulsory

* Git commit –a –m “”

-a adds all modified previously tracked files to staging area and then commits them

* git commit --amend -m "Initial Commit"

To correct the most recent commit message

***Amending the message of older or multiple commit messages***

*If you need to amend the message for multiple commits or an older commit, you can use interactive rebase, then force push to change the commit history.*

1. *On the command line, navigate to the repository that contains the commit you want to amend.*
2. *Use the git rebase -i HEAD~n command to display a list of the last n commits in your default text editor.*

*$ git rebase -i HEAD~3 # Displays a list of the last 3 commits on the current branch*

*The list will look similar to the following:*

*pick e499d89 Delete CNAME*

*pick 0c39034 Better README*

*pick f7fde4a Change the commit message but push the same commit.*

*# Rebase 9fdb3bd..f7fde4a onto 9fdb3bd*

*#*

*# Commands:*

*# p, pick = use commit*

*# r, reword = use commit, but edit the commit message*

*# e, edit = use commit, but stop for amending*

*# s, squash = use commit, but meld into previous commit*

*# f, fixup = like "squash", but discard this commit's log message*

*# x, exec = run command (the rest of the line) using shell*

*#*

*# These lines can be re-ordered; they are executed from top to bottom.*

*#*

*# If you remove a line here THAT COMMIT WILL BE LOST.*

*#*

*# However, if you remove everything, the rebase will be aborted.*

*#*

*# Note that empty commits are commented out*

1. *Replace pick with reword before each commit message you want to change.*
2. *pick e499d89 Delete CNAME*
3. *reword 0c39034 Better README*

*reword f7fde4a Change the commit message but push the same commit.*

1. *Save and close the commit list file.*
2. *In each resulting commit file, type the new commit message, save the file, and close it.*
3. *Force-push the amended commits.*

*$ git push --force*

* git push origin <feature branch name>

pushes changes from local repo to remote repo for feature branch

origin is reference used for remote repo

master branch to be pushes

* git branch

list all branches and highlight the current branch

* git branch <branchname>

just create a new branch do not switch to it

* git branch –d <branchname>

delete the branch, will show the unmerged changes before deleting

* git checkout <branchname>

switch to this branch

if the branch does not exist create it and switch to it

* git stash –u

keep all uncommitted files while keeping working directory clean

* git stash list

list all stashes

* git log

To see the commit history for the branch

To see file that was changed or added in a commit, use --stat argument with git log like this

* git log --stat.
* Git merge <source branch name> (The current branch should be the destination)

Merge changes of source branch with destination

After merge source and destination branch again are independent of each other, changes done on one do not reflect on the other

* Git revert <commit hexadecimal which you want to undo>

Git forward moving undo operation.

* Git rebase <source\_branch> (The current branch should be the destination)

Place commits of source branch on destination branch

Source branch remains unchanged

Now source and destination branch again are independent of each other, changes done on one do not reflect on the other

* git pull origin <feature branch name>

pull changes from remote repo referenced as origin to local repo for feature branch

* git remote –v

verify if a local repository is tracking the remote repository