Git Stash : Save’s un-commited work for later working purpose. So that you can work on something else and do some commits, later on you can unstash the files previously stashed and work on it.

Git stash	Will stash only Tracked & Staged files
Git stash -u	Will stash Untracked files, Tracked & Staged files
Git stash -a	Will Stash all files
Git stash **apply / pop**	Reverts back the stash, but now it will unstage the files if staged perviously.
Git stash list	List’s all the stash
Git stash drop / clear	Drops the stash OR clears the stash
Git stash save “message”	save’s the stash with the goven message
Git stash list Git stash show stash@{0}	shows the changes made in the file, same as “git show <commit-id>”
Git stash apply stash@{0} Git stash apply stash@{1}	apply’s OR reverts the stash 0 apply’s OR reverts the stash 1
Git stash drop stash@{**1**}	Drops the stash **1**
Git stash clear	Clears the stash
1. **Git stash -a** 2. **Git stash branch newBranch** 3. **Git checkout newBranch**	Stash all your work. Create’s a branch and save’s all the stash in it. Check the stash in this branch.

Git Tag : Used for tagging the commit as a **milestone/highlight**, it’s like a **lable/ref**. point to specific commit.
* d3d2227 (HEAD -> master, **tag: v1.0.0**) commiting changes made in app.py after stashing current work

Light-Weight Tag’s	
Git tag <tag_name> Git tag v1.0.0	Used to tag a commit, it will add the commit to the currently pointing HEAD if nothing is specified. * d3d2227 (HEAD -> master, **tag: v1.0.0**)
Git tag –list	Gives list of all tag’s in the current working repository.
Git tag –delete <tag_name> Git tag –delete v1.0.0	delete’s the tag. * d3d2227 (HEAD -> master)
Annotated Tag’s	**-a**
Git tag **-a** <tag_name> **-m** “message” Git show <tag_name> Git log –all –decorate –oneline **--**graph	Tags a commit with a annotated message. To check the message.
Git diff <tag_name1> <tag_name2>	Diffirence between the tags.
Git tag -a <tag_name> <commit_id> git tag -a **v1.0.0 c4bf5de** git show **v1.0.0**	Tags to that particular commit id * **c4bf5de (tag: v1.0.0**) Git intial commit.
Git tag -a <tag_name> **-f** <commit_id> git tag -a **v1.0.0 -f d3d2227** git show **v1.0.0**	If by mistake you set tag on wrong commit-id, this is used to correct it. * **d3d2227** (HEAD -> master, **tag: v1.0.0)**

Git Intial :

Git config –global user.name “sunil” Git config –global user.email “sunil.gurnale@gmail.com” Git config –global alias.history “log --oneline --decorate --graph --all” Git config –global –edit Create’s a file in ~/.gitconfig	Set Name Set Email Set Alias Opens file to edit changes Check this file
Git remote add <name> <URL> -- name for URL, which can be used later on Git remote add **origin** <git hub url> --instead of **origin** anything can be added Git push **origin** <branch name> -- origin is now shortform of the URL Git pull **origin** <branch name>	Add’s remote host to local repo.
Git init Git init <directory> Git clone <repo_url>	Initialize the git in the PWD Clones the repo from GitHub
Git branch <branch_name> Git branch developer Git branch -d developer	Create’s branch delete’s branch
Git checkout developer Git checkout master Git checkout <commit-id>	Switch to branch developer Switch to branch master Switch HEAD to that particular commit.
*master Git merge developer	Current location Master. Merge developer to master.
Fork on UI Git-hub	To make a copy of a repo owned by different owner in your github

Git Commands :

Git add . Git add <file_name/dir_name>	Add untracked files from current working directory to staging area.
Git commit <file_name> -m “message” Git commit -m “message” Git commit –amend 1. Git checkout <commit_id> 2. Git commit --amend	Commit particular file, from staging area to local repo. Commit all files in staged area to local repo. Change last commit message, for particular HEAD, give the commit-id in checkout.
Git push **origin** master/any branch name	Push all commits from LOCAL to REMOTE-Github repo
Git pull **origin** master	Always PULL before push
Git Status	Current Status of Git
Git log –all –decorate –oneline –graph Git log --stat	Detailed logs.
Git show <commit_id> Git show	Logs.
Git diff <1st branch> <2nd branch> Git diff <commit_id1> <commit_id2> Git diff –staged	Difference between different commit’s, branche’s etc
Git reflog --all --relative-date	Logs to local repo.’s HEAD

Git Additional :

Git clean -n Git clean -df	Shows which file can be removed from working dir. Cleans all the files from working dir. Untracked files: (use "git add <file>..." to include in what will be committed) fil11 fil12 fil13 fil14 fil15 root@nmap:/home/wifi1/Docker/git_practise# git clean -n Would remove fil11 Would remove fil12 Would remove fil13 Would remove fil14 Would remove fil15 root@nmap:/home/wifi1/Docker/git_practise# git clean -df Removing fil11 Removing fil12 Removing fil13 Removing fil14 Removing fil15 root@nmap:/home/wifi1/Docker/git_practise# git status

Git Ignore :

vim .gitignore

/logs/*	Git will ignore all the files in logs Directory
!logs/.gitkeep	Exculding the .gitkeep file
/tmp	Whole tmp directory ignored
*.swp	All .swp files ignored

github.com/github/gitignore