| **Command** | **Description** |
| --- | --- |
| git init | Initialize a local Git repository |
| git add -A | Add all new and changed files to the staging area |
| git commit -am “message” | Add and commit file at a time |
| git rm -r [file-name.txt] | Remove a file (or folder)/ If you want to remove the file from the Git repository **and the filesystem** |
| git rm --cached file1.txt | if you want to remove the file only from the Git repository and not remove it from the filesystem, |
| git branch | List branches (the asterisk denotes the current branch) |
| git branch -a | List all branches (local and remote) |
| git branch [branch name] | Create a new branch |
| git branch -d [branch name] | Delete a branch at Local |
| git push origin --delete [branch name] | Delete a remote branch |
|  |  |
| git checkout -b [branch name] | Create a new branch and switch to it |
| git checkout -b [branch name] origin/[branch name] | Clone a remote branch and switch to it |
| git branch -m [old branch name] [new branch name] | Rename a local branch |
| git checkout [branch name] | Switch to a branch |
| git checkout - | Switch to the branch last checked out |
| git checkout -- [file-name.txt] | Discard changes to a file |
| git merge [branch name] | Merge a branch into the active branch |
| git merge [source branch] [target branch] | Merge a branch into a target branch |
| git push origin [branch name] | Push a branch to your remote repository, do not remember the branch |
| git push --set -upstream origin <brachName> | Set branch name with upstream |
| git log | View changes |
| git log --summary | View changes (detailed) |
| git log --oneline | View changes (briefly) |
| git log origin/<branch\_name> --oneline | Log of origin branch |
| git log -2 | Last 2 log |
| git reflog | show full log, including rebase and reset commit |
| git diff [source branch] [target branch] | Preview changes before merging |
| git stash list |  |
| git stash -a/  git stash -u (--include-untracked) | Stashing all the files including tracked and untracked files and staged files |
| git stash clear | Remove all the stash entries. Note that those entries will then be subject to pruning, and may be impossible to recover (see Examples below for a possible strategy). |

**git stash apply**-> If you want to apply one of the older stashes, you can specify it by naming it, like this: git stash apply stash@{2}. If you don’t specify a stash, Git assumes the most recent stash and tries to apply it.

**git stash pop**

**git checkout** **-- file.txt** ->Undo changes for that particular file

**git checkout .** -> Undo changes for all the files

**git clean –xdf**

**git reset** -> Changes the Head to old commit, usually use in local, before committing to Remote to avoid conflict with other team member

**git reset HEAD@{1}**

3 types of reset is there – **Soft , Mixed, Hard**

git reset --hard <shaID of Commit>

**git revert**  -> add a new commit at the end, when the code has been pushed to Remote, helps to resolve conflicts, keeps proper history.

**git revert** **<sha1 of commit >** -> till that commit will be revert, actually one latest commit will be added with comparing instead moving the head to backword

**git rebase <branch\_name>** - > conflict occurred if there is a change in same file, we have to resolve conflicts file manually then **git add <file\_name> then git rebase --continue**

**git rebase --skip ->** to skip those for which conflict occurred

**git rebase --abort ->** aborting the rebase, move to the position where it was before

**git cherry-pick ->**

**git init -> create git repository at local machine**

**git remote add origin <url of the project>**