GIT COMMANDS


```
git show                                    # shows one or more objects (blobs, trees, tags and commits).
git diff                                    # show changes between commits, commit and working tree
git diff HEAD                               #show changes between working directory vs last commit
git diff --staged HEAD                      #show changes between stage area vs last commit

git diff --color                            # show colored diff
git diff --staged                           # Shows changes staged for commit

git tag                                     # shows all the tags
git tag -a v1.0 -m "msg"                    # creates an annotated tag
git show v1.0                               # shows the description of version-1.0 tag
git tag --delete v1.0                       # deletes the tag in local directory
git push --delete my-remote v1.0            # deletes the tag in my-remote (be carefore to not delete a branch)
git push my-remote my-branch v1.0           # push v1.0 tag to my-remote in my-branch
git fetch --tags                            # pulls the tags from remote

git pull my-remote my-branch                # pulls and tries to merge my-branch from my-remote to the current branch git pull = git fetch && get merge

git clean -f                                # clean untracked files permanently
git clean -f -d/git clean -fd               # To remove directories permanently
git clean -f -X/git clean -fX               # To remove ignored files permanently
git clean -f -x/git clean -fx               # To remove ignored and non-ignored files permanently
git clean -d --dry-run                      # shows what would be deleted
```

GIT COMMANDS


```
git log                                                  # shows the log of commits
git log --no-pager                                       # shows the log of commits without less command
git log --oneline                                        # shows the log of commits, each commit in a single line
git log --oneline --graph --decorate                     # shows the log of commits, each commit in a single line with graph
git log --since=<time>                                   # shows the log of commits since given time
git log -p <file_name>                                   # change over time for a specific file
git log <Branch1> ^<Branch2>                             # lists commit(s) in branch1 that are not in branch2
git log -n <x>                                           # lists the last x commits
git log -n <x> --oneline                                 # lists the last x commits, each commit in single line
git grep --heading --line-number '<string/regex>'        # Find lines matching the pattern in tracked files
git log --grep='<string/regex>'                          # Search Commit log

git reflog                                               # record when the tips of branches and other references were updated in the local repository.
git ls-files                                             # show information about files in the index and the working tree

git commit -m "msg"                                      # commit changes with a msg
git commit -m "title" -m "description"                   # commit changes with a title and description
git commit --amend                                       # combine staged changes with the previous commit, or edit the previous commit message without changing its snapshot
git commit --amend --no-edit                             # amends a commit without changing its commit message
git commit --amend --author='Author Name <email@address.com>'   # Amend the author of a commit
git push my-remote my-branch                             # pushes the commits to the my-remote in my-branch (does not push the tags)
git revert <commit-id>                                   # Undo a commit by creating a new commit
```

GIT COMMANDS

Pragyan Tripathi
@pragyanatvade

```
git init                                                    # initiates git in the current directory

git remote add origin https://github.com/repo_name.git      # add remote reposiory

git clone <address>                                         # creates a git repo from given address (get the address from your git-server) | git clone <address> -b <branch_name> <path/to/directory>

git clone <address> -b <branch_name> <path/to/directory>    # clones a git repo from the address into the given directory and checkout's the given branch

git clone <address> -b <branch_name> --single-branch       # Clones a single branch

git add <file_name>                                         # adds(stages) file.txt to the git

git add *                                                   # adds(stages) all new modifications, deletions, creations to the git

git reset file.txt                                          # Removes file.txt from the stage

git reset --hard                                            # Throws away all your uncommitted changes, hard reset files to HEAD

git reset --soft <commit_id>                                # moves the head pointer

git reset --mixed <commit_id>                               # moves the head pointer and then copies the files from the commit it is now pointing to the staging area,
                                                            # the default when no argument is provided

git reset -hard <commit_id>                                 # moves the head pointer and then copies the files from the commit it is now pointing to the staging area
                                                            # and working directory thus, throw away all uncommitted changes

git rm file.txt                                             # removes file.txt both from git and file system

git rm --cached file.txt                                    # only removes file.txt both from git index

git status                                                  # shows the modifications and stuff that are not staged yet

git branch                                                  # shows all the branches (current branch is shown with a star)

git branch -a                                               # shows all the branches local and remote

git cherry-pick <commit_id>                                 # merge the specified commit

git cherry-pick <commit_id_A>^..<commit_id_B>               # pick the entire range of commits where A is older than B ( the ^ is for including A as well )
```

GIT COMMANDS


```
| git config --global --list                                  # lists the git configuration for all repos
| git config --global --edit                                  # opens an editor to edit the git config file
| git config --global alias.<handle> <command>                # add git aliases to speed up workflow , eg.
| git config --global core.editor <editor_name>               # config default editor
| git archive <branch_name> --format=zip --outpute=./<archive_name>.zip   # create an archive of files from a named tree

| git stash                                                   # stashes the staged and unstaged changes (git status will be clean after it)
| git stash -u                                                # stash everything including new untracked files (but not .gitignore)
| git stash save "msg"                                        # stash with a msg
| git stash list                                              # list all stashes
| git stash pop                                               # delete the recent stash and applies it
| git stash pop stash@{2}                                     # delete the {2} stash and applies it
| git stash show                                              # shows the description of stash
|git stash apply                                              # keep the stash and applies it to the git
| git stash branch my-branch stash@{1}                        # creates a branch from your stash
| git stash drop stash@{1}                                    # deletes the {1} stash
| git stash clear                                             # clears all the stash

| git rebase -i <commit_id>                                   # Rebase commits from a commit ID
| git rebase --abort                                          # Abort a running rebase
| git rebase --abort                                          # Continue rebasing after fixing all conflicts
```