```
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>  # 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.txt   # 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 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 my-branch                # creates my-branch
git branch -d my-branch             # deletes my-branch
git checkout my-branch              # switches to my-branch
git merge my-branch                 # merges my-branch to current branch
git push origin --delete my-branch # delete remote branch
git branch -m <new-branch-name>    # rename the branch
git checkout --orphan <branch_name> # checkout a branch with no commit history
git branch -vv                      # list all branches and their upstreams, as well as
last commit on branch
git branch -a                       # List all local and remote branches

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 remote                          # shows the remotes
git remote -v                       # shows the remote for pull and push
git remote add my-remote <address> # creates a remote (get the address from your git-
server)
git remote rm my-remote             # Remove a remote

git log                       # shows the log of commits
git log --oneline             # shows the log of commits, each commit in a single line
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 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 show                     # shows one or more objects (blobs, trees, tags and
commits).
git diff                     # show changes between commits, commit and working tree
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 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 --continue             # Continue rebasing after fixing all conflicts

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 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.
if  handle is st and command is status then running git st would execute git status


.gitignore
# is a file including names of stuff that you don"t want to be staged or tracked.
# You usually keep your local files like database, media, and etc here.
# You can find good resources online about ignoring specific files in your project files.
# .gitignore is also get ignored
.git
# is a hidden directory in repo directory including git files. It is created after "git
init".
```