Git

- git config --global user.name “Piyush Kalantri” → to set user name
- git config --global user.email “piyush10001@gmail.com” → to set user email
- git config --list → to see user details
- git help <verb> / git <verb> --help → to need help with any of the actions(eg. git help config / git config --help)
- git init → to initialise an empty git repo.
- rm -rf .git → to remove git repo / to stop tracking the project with git
- git status → to check which files are tracked and which are not
- touch .gitignore → will create .gitignore file in which state the files which you want git to ignore
- git add <file> → to add <file> to staging area
- git add -A / git add . → to add all files to staging area, irrespective of the directory we are in
- git reset <file> → to remove <file> from staging area
- git reset → to remove all files from staging area
- git commit -m “message” → to commit files to repo with a message
- git log → to check history of commits
- git clone <url> <where to clone> → to clone the repo which is at the url to your directory (eg. git clone ../repo.git .), here “.” means current directory
- git branch → shows all the local branches
- git branch -a → shows all the local as well as remote branches
- git remote -v → shows info about the repo
- git diff → shows the difference between the remote repo and the local files
- git pull <repoName> <branchName> → to pull latest commits from the remote branch to our local branch
- git push <repoName> <branchName> → to push latest commits to the remote branch from our local branch
- git branch <branchName> → to create a new branch
- git checkout <branchName> → to shift to checkout branch
- git push -u <b1> <b2> → to push commits from b2 to b1 (-u is used to associate b2 to b1, so that next time we do “git push” from b2 branch it pushes into b1 branch.
- git branch —merged → shows all the branches which are merged to the current branch
- git merge <b> → to merge branch <b> to the current branch
- git branch -d <b> → to delete branch <b> locally
- git branch <repoName> --delete <b> → to delete branch <b> from repo
- git checkout <fileName> → to load the file from repo, so that the changes

you made will disappear

- git commit --amend -m “message” → to change the message for the last commit
- git commit --amend → to add the changes made at the staging area to the latest commit
- git log --stat → will show the files that were changed within the commit
- git cherry-pick <hash> → to add a commit <hash> to another branch
- git reset --soft <hash> → remove the commits after <hash> commit but the changes remain in the staging area
- git reset <hash> → removes the commits after <hash> commit but the changes are in the working directory not in staging area.
- git reset —hard <hash> → roll back the tracked files to the <hash> commit , but the untracked files will be in the working directory , not in staging area.
- git clean -df → to remove untracked directories(-d) & to remove untracked files(-f)
- git reflog → shows the commits in the order of when you last referenced them
- git checkout <hash> → creates a branch(temp) which will store all commits till <hash>, create a backup branch after this command , to store the commits in that branch till <hash> commit
- git revert <hash> → to undo the <hash> commit , it will create another commit which will undo the changes made in the <hash> commit
- git diff <hash1> <hash2> → shows the difference between the 2 commits
- git stash save “message” → saves the changes made in the working directory(sort of a stack), use when you are not sure of the changes you have made
- git stash list → shows all the stash
- git stash apply <stash@{0}> → to apply the changes made in <stash@{0}> to the current branch, the stash will still remain
- git stash pop → will apply the changes made in the last stash and remove the stash
- git checkout -- . → (iska meaning theek se nahi samjha) goes back to the start
- git stash drop <stash@{0}> → deletes the stash, used when the changes made in the stash are not required
- git stash clear → deletes all stash
- stash is common for all branches.
- diff & merge tools (like CR), shows the changes made in the specific line
- git add -A <dir> / git add <dir> → will only stage all changes made to <dir> directory
- git add -u / git add —update → will stage only modified and deleted files, wont stage any untracked files
- git add -u <dir> → same as above, but only the changes made to the <dir>
- git add . → will stage all the changes within the current directory

- git pull --rebase → will pull the commits from remote repo and put your commit on top