Git & Github (Using git bash)

commands used in git :-

`git config --global user.name Harshita`

↓ give your name to git bash

`git config --global user.email "..."`

↓ for giving email add.

`code .` → to open VS code

↓ space

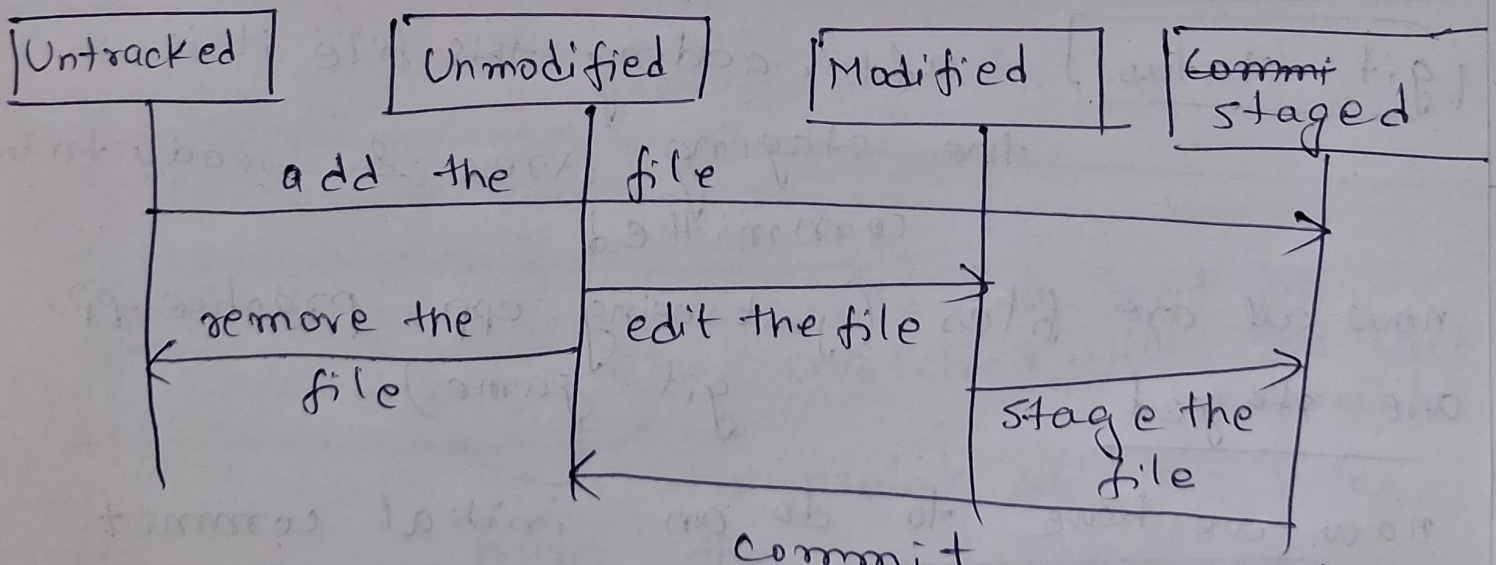
Initialisation of a ~~file~~ Repo → formed an empty Repo

`git init` → initialized empty Git repo

`ls` → all folders

`ls -lart` → all folders including hidden ones

concepts



Untracked: git has got nothing to do with these types of file.

Staged: we add files from untracked, edit the files from unmodified to modified which is then staged) we add all the files to the staging area which is to be committed.

Unmodified: (snapshot by Git)
↳ files which has be commited, these files can be either left unchanged or modified or removed.

modified: files which are modified has to be then staged. ⟶ commit

---

We have some untracked files:

`git add index.cpp` or justseeing.cpp

`git status` after adding this file it is in the staging area & ready to be committed

now all are files (justseeing.cpp, practice.cpp, git-game) are staged

now we have to do an initial commit

`git commit`

Vim terminal will open (Type Initial Commit :wq)

all files are commited Now

to create blank files touch

```
touch about.cpp
touch contact.cpp
touch mon.cpp
```

git status will show these as untracked files

`git add -A` → to add all the files at once from untracked to staging area

`git add filename.txt` → to add a specific file

`git add *.js` or cpp → to add all files with a specific extension.

to commit

`git commit -m "Add a messege"`

↓

shorter way to commit

`git checkout justseeing.cpp` → this file will not be matched with the last commit

modified & will be

`git checkout -f` → to match all the files from the last commit

`git log` → gives info (commit history)

`git log --oneline` → compact form

`git log --graph --oneline --all` → to view graphical history

`git log -p -n` → n: to see the history of last n commits

`git diff` → compares my working directory with the staging area & points out the difference (if any).

`git diff --staged` → See what will be committed

↓

to compare the staging area from the last commit.

---

to commit without going to the staging area

`git commit -a -m "$kippeds"`

`ls` → listing of all files

`git rm git-game` → remove this file from the working directory & the staging area

git rm --cached → will be removed just from the staging area (will not be deleted from the harddisk)

name

file will go to the list of untracked files

---

git status -s → compact info of all files (M, S, U)

.gitignore (Imp) [we sometimes need to ignore some files like log, cache etc]

so if we make/have many files & write there names in the file of .git ignore then those files will not be visible when we run any command on git bash.

.gitignore → will ignore this if it's in .gitignore

(Im) 4 logs.log

*.log → ignore all log files in repo

logs/ → ignore this directory called logs

folder

`git branch feature1` → creation of a branch called feature1

`git branch` → list of all Branches (local)

main
feature1 → currently we are in main

`git checkout feature1`

now we are in ~~feature1 branch~~

`git merge feature1` → feature1 branch got meaged with master

`git checkout -b feature1`

this branched is formed as well as we switched to this Branch

`git branch -a` → List of all branches including remote & local

`git branch --show-current`

see current branch

`git switch -c "branch name"`

upgraded way

git checkout branch-name

||

git switch branch-name

---

git branch -d branch name → to delete local branches before merging.

git branch -D branch_name → forced delete before merge

git push origin --delete Branch-name → delete remore branchs

Pushing (UpLoading local commits to remote repo)

git push → Push current branch

git push origin branch-name → Push to remote Repo

git push -u origin branch name → push & set upstream

Pulling from remote repo:

git pull → fetch and merge changes

git fetch → fetch without merging

git pull origin main → Pull from specific branch

Merging :

① Fast forward merge
- when there is no new commits on the target branch
- No merge commit is created
- Linear history

```
git checkout main
git merge new-branch
```

② 3-ways merge:
- Used when both branches have new commit
- git creates new branch commit
- Preserves both branch history

```
git checkout main
git merge new-branch
```

③ Squash merge
- combines all commits into one
- creates a ~~sign~~ single commit on target branch
- loses commit history

```
git merge --squash new branch
git commit -m "Merged"
```

Basic merge workflow :

make sure main is up to date

```
git checkout main
git pull origin main
git merge new-branch
git push origin main
```

Viewing Merge History:

`git log --merges` → See merge commit

`git log --graph --oneline --all` → Graphical view of merges