**Git basics**

After installing git ,

Once we open a project in code we type

“Git init “ command to initialise the git repository.

Repository is like a container.

.git folder will be created after we do this.

**Several phases in a file**

Tracked or untracked

Tracked: Changes which were in last snapshot

Untracked: Changes which were not in last snapshot

Git add —> before committing u do this to stage the changes or new files.

Git rm —cached —> to unstage if we want.

Git commit -m “” —> to commit the changes.

Git remote add origin {repo link} —> to link our local repo to remote repo

Git push origin master —> push the changes to remote

Git pull origin master —> to pull the latest changes from the remote

Git config —global user.name

Git config —global user.email

We need to give an username and email or set up an identity initially.

Git log / git log —oneline —> gives the history of the commits.

**How to undo or discard things in git?**

Head —> is a pointer ,points to a default branch, points to the last commit.

To unmodify files

Git checkout

Git checkout {id of the commit} —> it will go back to previous state of the project

But if we want back the current state again

Git checkout {branchname}.

Git revert {id of the commit} 🡪 reverts the changes from the last commit or can undo last commit

Git reset —{flags} {id of the commit} —> dangerous command coz can delete the work permanently

Has three flags

1. Mixed(default) —> removes the commits from the history and also unstages the files from the staging area.
2. Soft —> here the files are not unstaged
3. Hard.

**Difference between revert,reset and checkout**

<https://www.geeksforgeeks.org/git-difference-between-git-revert-checkout-and-reset/>

.gitignore —>ignores files

Origin- name of the remote

**Branching :**

Git branch {branch name} —> to create a new branch

Git checkout {branchname} —> to switch to different branch

Git branch -D {branch name} —> to delete a branch

To merge 2 branches

1. Fast forward —> master gets overrided

Git merge {branchname}

1. 3-way merge —> if we want both the branch changes

**Merge conflict—>**

In remote

If there are 2 branches in remote with some commits

To merge 2 branches

1. Create pull request

It will automatically merge if there are no conflicts

But if 2 branches are trying to modify the same line of code then what to do???

Merge conflict arises

“Can’t automatically merge” —> error happens

So even if we press pull request button ,some warnings arises saying to resolve conflicts.

When we press the resolve conflicts button it takes us to the editor where we need to edit our code which is causing the conflicts and finally press “mark as resolved” button

Commit the merge and then merge pull request —> successfully it gets merged

**Forking and cloning**

Fork —> if we want to work on someones project we can copy the entire repository of theirs to our account by just clicking “fork”

Clone —> to clone a project

git clone {projectlink}

**Contribute on original projects**

If we want to contribute on original projects we can update the repository and add Our changes and then send a pull request to the owner and only when the owner approves the changes the changes will get pushed to the original repository

**Collaboration**

Owner can add different collaborators to the project so that they can work together on the same project.