Git

→ Version Control System is a tools that helps to track changes in code.

Example : It is similar to a Banking System that stores records and information about all the transactions.

→ Git is a Version Control System.

→ It is :

* Popular
* Open Source & Free
* Fast & Scalable.

→ It

* Tracks the history of Code.
* helps us to Collaborate.

→ Git is a Tool / Software that runs on our local system.

GitHub

→ Website that allows developers to store & manage their code using Git.

https://github.com

Readme File

→ It is stored as README.md.

→ .md stands for markdown.

→ It

* Tracks the history of Code.
* helps us to Collaborate.

→ Git is a Tool / Software that runs on our local system.

GitHub

→ Website that allows developers to store & manage their code using Git.

https://github.com

Readme File

→ It is stored as README.md.

→ .md stands for markdown.

→ Readme file is different from normal text editors.

→ It uses different syntax to write content into it.

→ To write contents into readme file we can also use HTML syntax.

Setting up Git

→ Windows (Git Bash)

→ Mac (Terminal)

→ Visual Studio Code (VS Code)

- To download Git go to

  https://git-scm.com/download

- To check

```
$ git --version
```

Configuring Git

```
$ git config --global user.name "Name"
```

```
$ git config --global user.email "abc@email.com"
```

```
$ git config --list
```

↑

This command is used to check what configurations we have done in git.

It provides list of all configurations done.

Git Commands

Clone - Cloning a repo on our local machine.

`$ git clone <-- ssome link -->`

Status - displays the state of the code

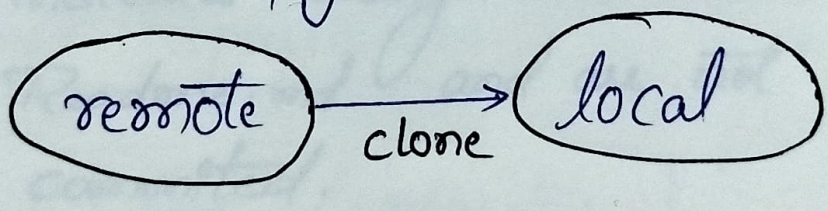
`$ git status`

Here the words refer to

→ remote [GitHub]

→ local [laptop / PC]

* Clone to copy from



⇒ We use the following commands in terminal to gain more insights

`$ cd` → change directory

`$ ls` → list files (excluding hidden files)

`$ ls -a` → list all files along with hidden

`$ clear` ⟶ to clear the terminal

Note :-

* After making changes we must always i) add
ii) commit
the changes made to a file.

* To check whether you have done the above steps use git status

* In status

modified : Readme.md

⟶ indicates changes are made to Readme.md and are not commited.

Untracked files :

⟶ indicates new files are created which git previously did not recognize / know.

Status Types in Git

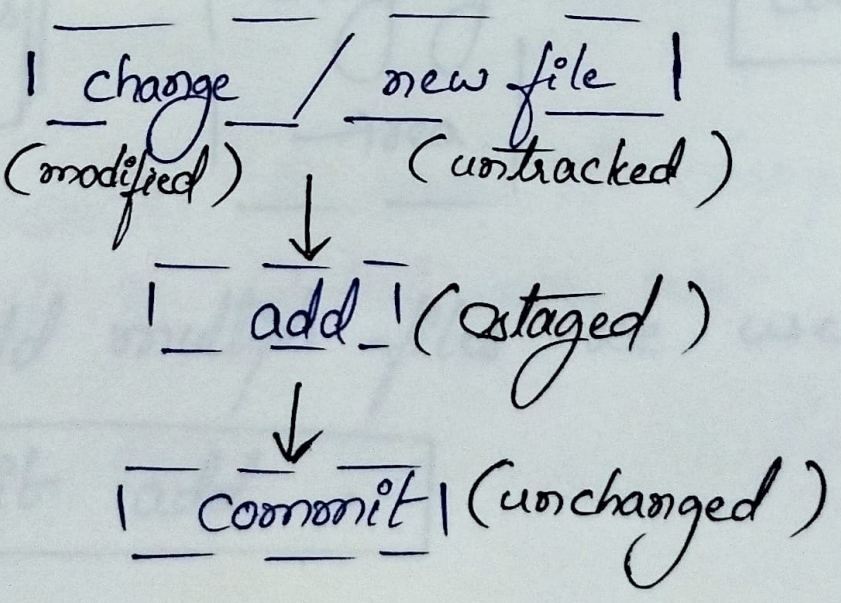
→ **Untracked:**
new files that git doesn't yet track.

→ **Modified:**
changed

→ **Staged:**
file is ready to be committed

→ **unmodified:**
unchanged

Flowchart Representation



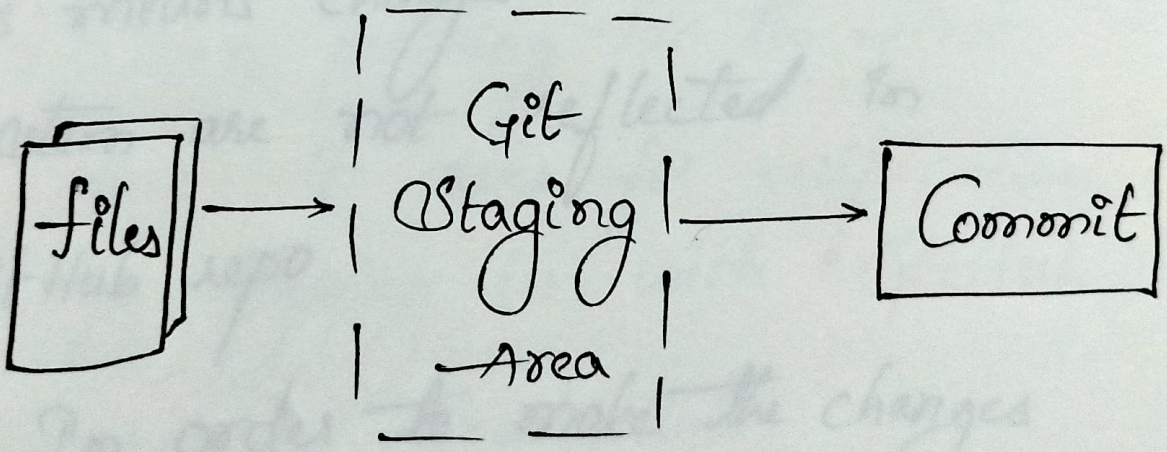
*

⇒ Git tracks all the changes when you add & commit.

Add & Commit

add – adds new or changed files in your working directory to the Git Staging area

```
$ git add <- file name ->
```



• To add multiple files we use

```
$ git add .
```

commit — It is the record of change.

```
$ git commit -m "some msg"
```

*⇒ After commit when you check status it displays.

```
Your branch is ahead of 'origin/main'
by 1 commit.
```

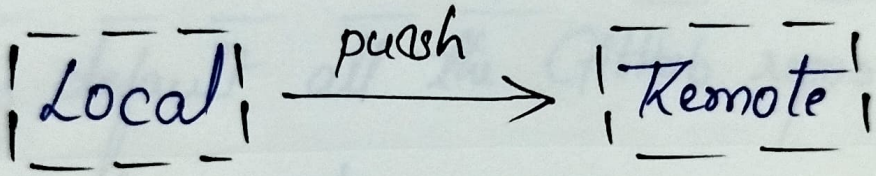
This means changes made in local systems are not reflected in GitHub repo

*⇒ In order to make the changes reflect in GitHub we use push command.

Push Command

→ push — upload local repo content to remote repo.

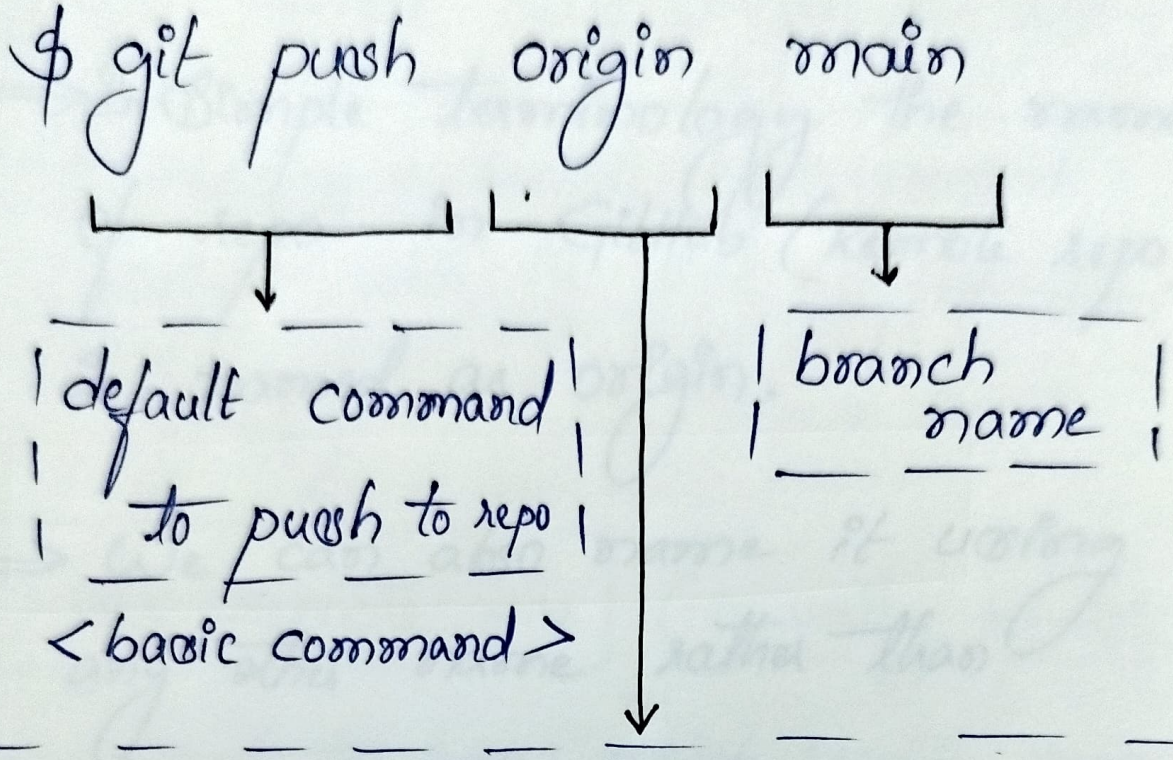
```
$ git push origin main
```



* ⇒ When pushing for the first time it asks for verification by logging in with GitHub Account Credentials.

* ⇒ After pushing changes will reflect in your GitHub repo.

Explaination



⇒ By default all the GitHub repos are called remote repos

⇒ From all the repos we choose a specific Remote repo which is a default repo from which we have cloned the code

⇒ On the default repo on which we make changes that should reflect on remote repo

which named origin, when pushed.

⇒ In simple terminology the name of repo in GitHub (Remote repo) is named as origin.

⇒ We can also name it using any other name rather than origin.

⇒* To manage the files on local system and reflect them on github we use init command.

Init Command

init – used to create a new git repo.

`$ git init`

`$ git remote add origin <- link ->`

→ Adding a remote repo named origin with its link (copied from GitHub)

`$ git remote -v` (to verify remote)

→ Used to know which remote repo we are set to. (i.e on which repo we perform the push operation)

`$ git branch` (to check branch)

→ In olden days by default the branch was "master" but later GitHub changed its policies now the default branch in GitHub

is "main".

⟹ But be careful while using git as default branch in git is still master and now master and main are two different branches and are not same.

```
$ git branch -M main
```

(to rename branch)

"new name" of branch

```
$ git push origin main
```

⟶ To push the commited changes to remote repo.

*

⇒ When you are working for a long time on same project instead of typing "origin main" everytime you push the changes you can set an upstream (i.e next time you push the changes all the changes are pushed to the same repo.)

⇒ Command for that is (Setting an upstream)

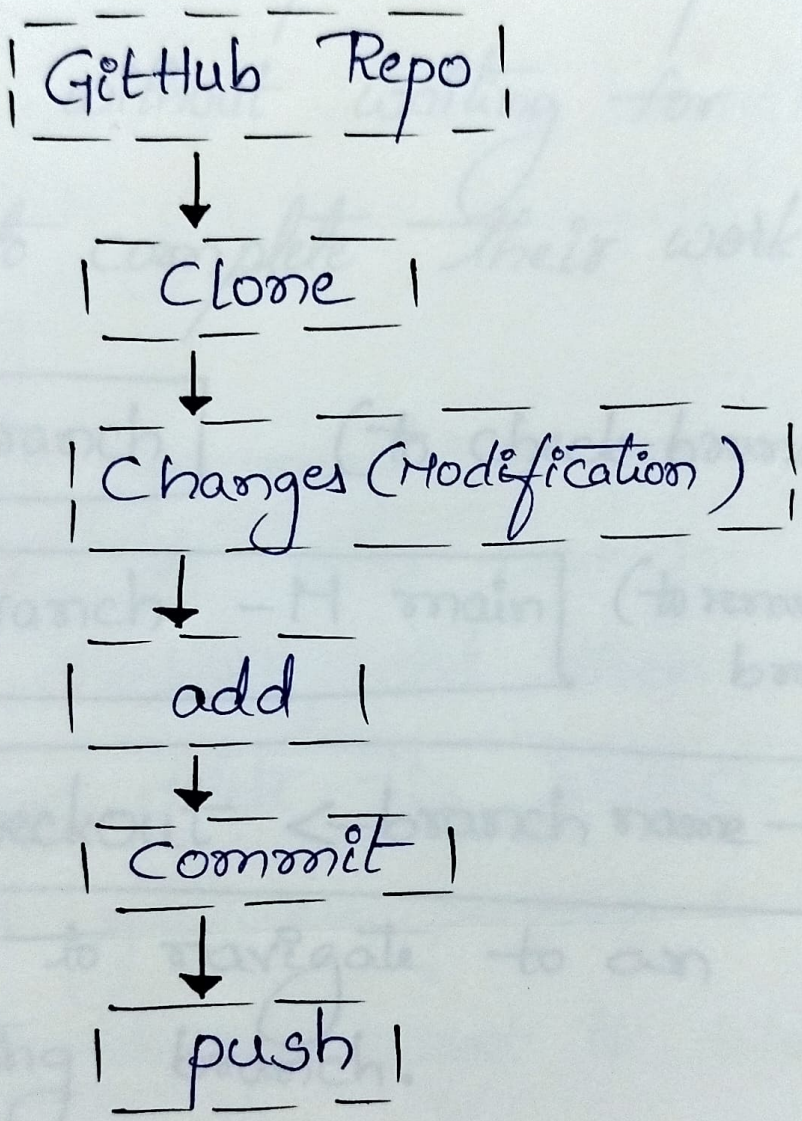
```
$ git push -u origin main
```

Setting an upstream

⇒ From next time from now you can use only git push changes are always pushed to same repo until

changed / changing the repo you want to push to.

WorkFlow (Local Git)



Branches

→ They are created so multiple people can complete their part of work without waiting for others to complete their work

```
$ git branch
```
(to check branch)

```
$ git branch -M main
```
(to rename branch)

```
$ git checkout <-branch name->
```

→ used to navigate to an existing branch.

```
$ git checkout -b <-new branch name->
```

→ used to create a new branch, that does not exist.

$ git branch -d <-branch name->

→ used to delete a branch, which is existing.

Note :-

→ You cannot delete a branch in which you are currently working.

→ First you need to move/ navigate to other branch using checkout and then you can delete the branch (only if you were present in the branch you want to delete).

→ If you try to delete branch in which you are present / currently working it displays an error.

Merging Code Branches

Method - I

Using Git

```
$ git diff <- branch name ->
```

→ used to compare commits, branches, files & more.

```
$ git merge <- branch name ->
```

→ used to merge 2 branches.

Method - II

Using GitHub

→ Create a PR (Pull Request).

Pull Request

→ It lets you tell others about changes you've pushed to a branch in a repository on GitHub.

→ This pull request will be reviewed by the senior developer and comments on them to let us know about the changes.

→ If accepted, then senior developer will merge the branches.

* ⟹ When we want to move the changes from remote to local we use pull command.

Pull Command

```
$ git pull origin main
```

→ used to fetch and download content from a remote repo and immediately update the local repo to match that content.

Resolving Merge Conflicts

→ An event that takes place when Git is unable to automatically resolve differences in code between two commits.

→ We need to resolve it manually.

Undoing Changes

Case 1 : Staged Changes (add but not committed)

`$ git reset <- file name ->` (single file)

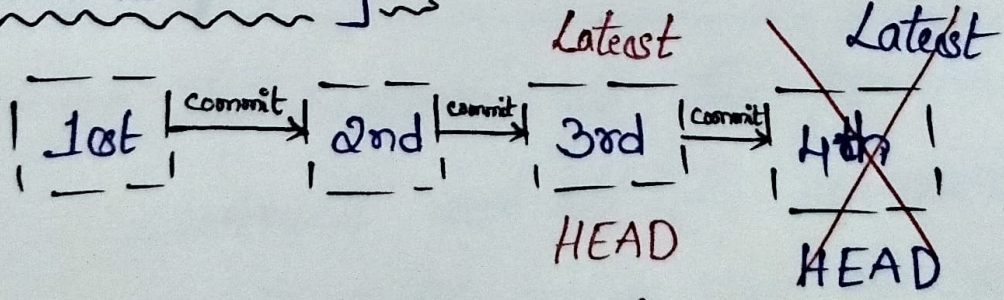
`$ git reset`

↳ used to reset all the files

Case 2 : Commited Changes (for one commit)

`$ git reset HEAD~1`

Commit History



`HEAD~1` → <reverts back by 1 commit>

* ⟹ To check all the commits made in terminal we use

```
$ git log
```

Note:-
To exit from log hit 'q'.

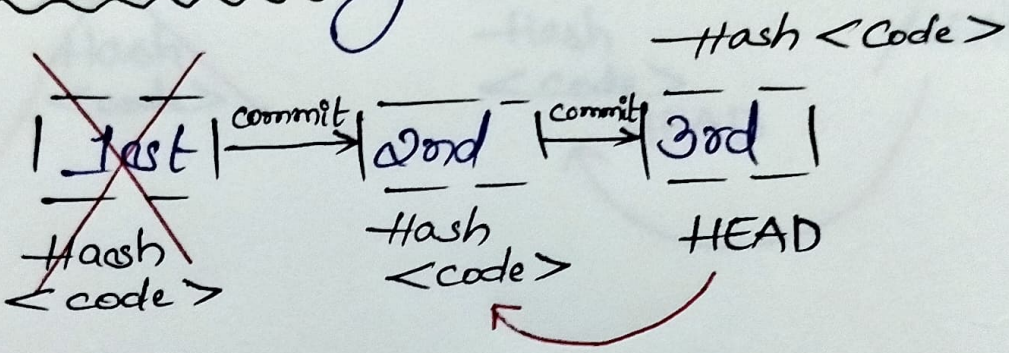
⟶ All the commits will be displayed from recent ⋮ old. } In this order.

⟶ Every commit is associated with a unique hash code.

Case 3 : Commited Changes (for many commits)

```
$ git reset <- commit hash ->
```

Commit history



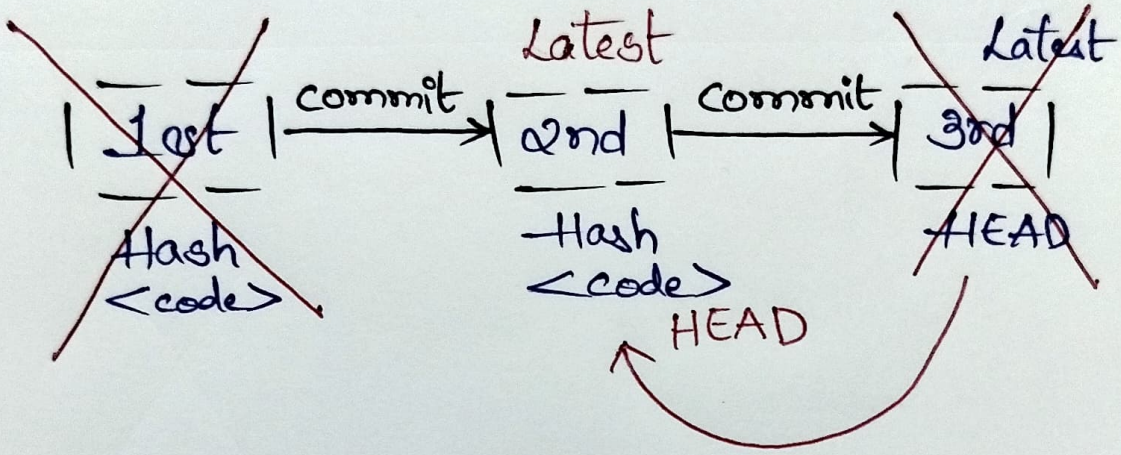
→ This removes all the previous commits made, from git but not from VS Code Editor.

```
$ git reset --hard <-commit hash->
```

→ This removes all the previous commits and all the commits made after the current commit in which you are positioned both

from git and VS Code Editor.

Commit History



Fork

→ A fork is a new repository that shares code and visibility settings with the original "upstream" repository.

→ Fork is a rough copy.