Notes- Learning Interface b/w Git and Github

GIT
- → What? → Distributed version control system.
- → Why? → To keep track of each version for software developers
- → How? → Creates snapshot of project files at a particular time.
- → Who? → Linus Torvald
- → When? → 2005
- → Where? → operates locally

GITHUB
- → What? → Platform that hosts git repositories
- → Why? → Hosting repositories, collaborating, visibility
- → How? → Github provides a platform for coders to store their repo.
- → Who? → Tom Preston-Werner, Chris Wanstrath and PJ Hyett
- → When? → 2008
- → Where? → Operates remotely

Basic linux commands (Only the used ones listed)

→ ls -a → Shows the files (including hidden ones)
→ cd <path> → Allocates path
→ cd .. → Moves one step up in folder hierarchy
→ mkdir <new directory name> → Create new directory
→ clear → clear screen

Local Code → Github

Write the code in your local code editor (say vs code)

↓

① Convert into git file

↓

② Add files (staging)

↓

③ Commit file

↓

④ push file

Github code → Local

Create a folder in your local where you import the code

↓

① Clone the repo in your local

↓

② Make the modifications in your local and save the file

↓

Follow the steps of local to github. Be careful with the branch where you want to push the code.

---

① Convert into git file

Commands to remember:

→ ls -a → check status

→ git init → .git conversion

② Staging (Add files)

→ git status → current status of files

→ git add<filename> → Adds the file

→ git add . → Adds all the files in the folder

③ Commit file

→ git commit -m < Write commit changes here >

④ push file

→ git remote add origin < link to repo >

→ git push origin main

(pushes to branch main)

(alias origin created for link)

① Clone the repo

git clone < link to repo >

② Make code modifications and save the file

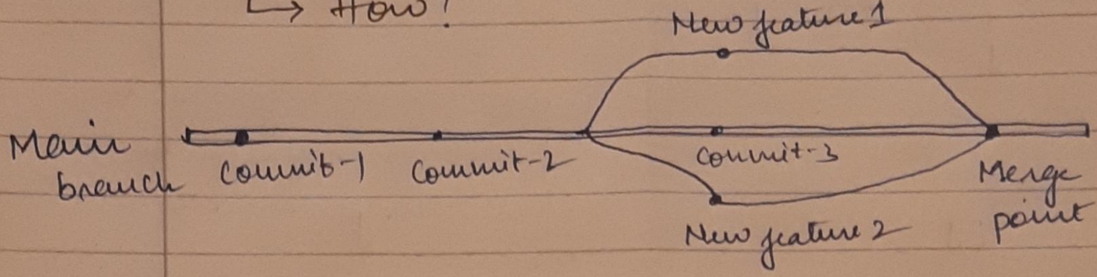
Be careful with the branch before pushing

Branching

→ What? Create multiple divergent paths of development in a project

→ Why? Branches isolate changes (parallel processing)
Each developer can work independently
Reduces the risk of errors in the main code

→ How?



→ git branch → To check my current branch (in gitbash you can see in brackets)

→ git branch -M <new branch name> → To rename a branch

→ git checkout <branchname> → To navigate to a branch.

→ git branch -d <branch name> → To delete a branch (you can't delete the branch where you are present currently)

→ git checkout ~~-b~~ -m <new branch name> → To create a new branch

Implementation of branch

step 1 → I opened my file GitHubToLocal.ipynb and added few lines of code and saved the file

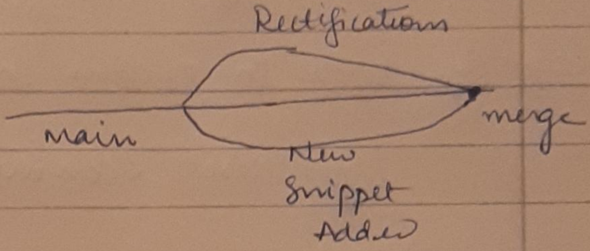
step 2 → In gitbash terminal

```
git branch -m Rectifications → (created a new branch rectifications)
git checkout Rectifications → (moved to branch rectifications)
git add . → staging
git commit -m "Adding changes to branch rectifications" (committed some changes)
git remote add origin <link>
git push origin Rectifications
→ push
```

This creates a branch Rectifications which is one commit ahed of the main branch

→ Similarly I created ~~a new~~ another new branch "New Snippet Add"

<u>Merging Branches</u>



I Way

To merge

First check differences using
```
git diff <branch name>
```

Then merge → `git merge <branch name>`

II Way

By creating a <u>pull request</u> (PR)

→ It lets you tell others about the changes you have pushed to a branch in a repo on github.

PR is reviewed by senior developer

Branch
↓
main

Review any conflicts and complete the merge.

Once the merge is done through PR — to bring the changes in local use → `git pull origin main`

Unstaging

git reset <filename>

git reset

Undoing latest commit

→ git reset Head~1 (Brings back to unstaged level)

→ Use→ git log (to check the log book of commits)
shows hashcodes

→ Go back to certain commit

git reset <commit hash> (Copy paste the hashcode from git log)

Forking Y

~~esa~~ Creating a rough copy in our repository of others projects is called forking.

We can fork an open source project — add/contribute valuable changes — create pull request