Git

* Git = Free and open source Version Control System

VCS = Version Control System = Tools that helps to tracks chages in code

* Github = Website where we host repositories online

* Using Git

→ Command Line
→ IDE / Code Editors (like VS code)
→ Graphical User Interface (like GitKraken)

* Configuring Git

(We are using git bash)

→ git config --global user.name "italiyaparth"
→ git config --global user.email "parthitaliya@gmail.com"

→ git config --list ⇐ information of our git account

```
:
init.defaultbranch = main
user.name = italiyaparth
user.email = parthitaliya@gmail.com
init.defaultbranch = master
:
```

* Terminal in VSCode

→ press ctrl+j or stretch the bottom line, a panel will be seen which has some tabs, click on TERMINAL tab

→ In default setting, terminal will be in powershell mode change it to git bash.

* Clone a Repositories to local machine

→ clone = making a copy

→ On Online github, make a repository named test (when making it check on README file).

→ After that, github will ask you to upload code, .etc. There is a code button in green color, clicking on it, we will see clone option HTTPS, SSH, GitHub CLI.

→ Copy HTTPS link

→ Now, create ~~test~~ demo named folder on desktop and we will clone our repository in this demo folder in local machine

→ pwd
/c/Users/HP

→ ~~cd Downlo~~
→ cd Desktop
HP@DP MINGW64 ~/Desktop
$

→ mkdir demo

→ cd demo
HP@DP MINGW64 ~/Desktop/demo
$

→ git clone https://github.com/italiyaparth/test.git
There will be test folder inside demo folder as output

→ cd test
HP@DP MINGW64 ~/Desktop/demo/test
$

→ ls -la
total 5
drwxr-xr-x 1 HP 197121 0 sep 6 22:58 ./
drwxr-xr-x 1 HP 197121 0 sep 6 22:54 ../
drwxr-xr-x 1 HP 197121 0 sep 6 22:54 .git/
-rw-r--r-- 1 HP 197121 6 sep 6 22:59 README.md

→ touch index.html style.css ← two new files created
→ code index.html style.css README.md ← three files open in VS Code

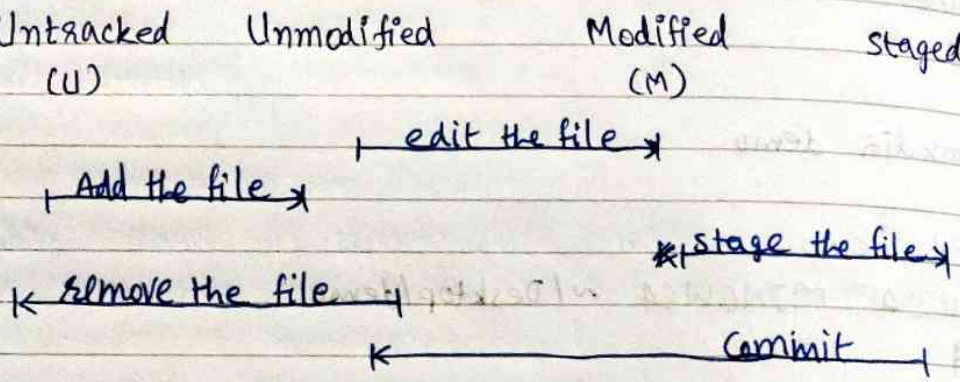
↑ names are case sensitive

→ Add some lines in README.md files, save it

* Status

→ status = Displays the state of the code

File Status Lifecycle



→ git status

```
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (Use "git add <file>..." to update what will be committed)
  (Use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (Use "git add <file>..." to include in what will be committed)
        index.html
        style.css
```

no changes added to commit (Use "~~git~~ git add" and/or "git commit -a")

* Add & Commit (cloned Local to online (remote))

→ add = adds new or changed files in your working directory to the Git staging area

→ commit = It is the record of change

→ push = Upload the local repo content to remote repo

→ git add index.html

→ git status

On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
(Use "git restore --staged <file>..." to unstage)
new file: index.html

Changes not staged for commit:
(Use "git add <file>..." to update what will be committed)
(Use "git restore <file>..." to discard changes in working directory)
modified: README.md

Untracked files:

(Use "git add <file>..." to include in what will be committed)

style.css

→ git add . ← means all files

→ git status

On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
(Use "git restore --staged <file>..." to unstage)
modified: README.md
new file: index.html
new file: style.css

→ git commit -m "Added new files"

→ git status

On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(Use "git push" to publish your local commits)

nothing to commit, working tree clean.

→ git push origin main
→ Now, You can see after refreshing github online Updates will be there

* New created Repo on Local machine to Github Remote

→ Make a new directory in demo folder : test2

- cd ~
- cd Desktop/demo
- mkdir test2
- cd ~
- cd Desktop/demo/test2
- ls -la

  ..../
  ... .../

→ init = Used to make (create) a new git repo

- git init

  ~~git~~

- ls -la

  ..../
  ... .../
  ... ... .git/

- touch index.html README.md
- code index.html README.md

→ write some code in html and README.md

- git add .
- git status

On branch master

No commits yet

changes to be committed
(Use git rm --cached <file>..." to unstaged)
new file: README.md
new file: index.html

- git commit -m "first commit"
- git status

On branch master
nothing to commit, working tree clean

- git push origin main
error: src refspec main does not match
error: failed to push some refs to 'origin'

→ Make new repo on github online with same name as folder name ~~of~~ of local machine
copy link of repo HTTPS

- git remote add origin https://github.com/italizaparth/test2

- git remote -v ← verify remote
origin https://github.com/italizaparth/test2.git (fetch)
origin https://github.com/italizaparth/test2.git (push)

- git branch ← check branch
* master

= git branch -M main ← Rename branch

- git branch ← check branch
  * main

- git push origin main

→ Verify on github online (new files will be there)

→ Now, add something in README.md

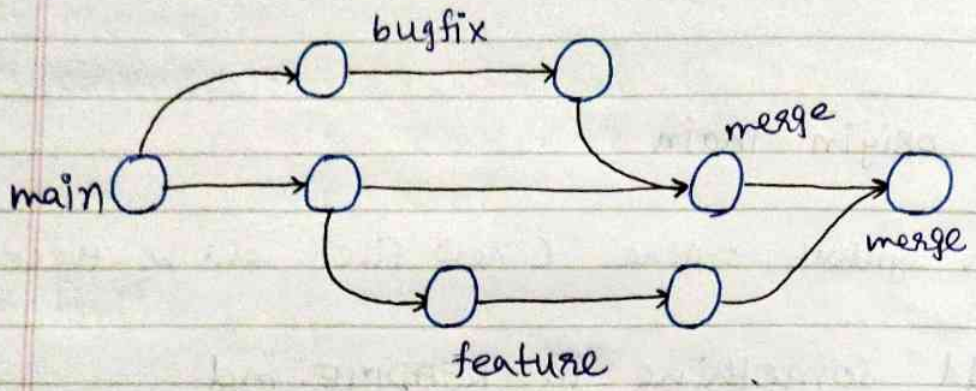
- git add .
- git commit -m "added something"
- git push origin main

→ Verify on github online (files will be updated)

Note: If one time we give (upstream) this command
git push -u origin main
then for push we can write this:
git push

Note: If No new files created and only some changes happened in the file then we can write this:
git commit -am "update some changes"

* Git Branches



→ git branch = To check branches
→ git branch -M main = To rename branch
→ git checkout branchName = To Navigate
→ git checkout -b branchName = To create a new branch
→ git branch -d branchName = To delete a branch

- git checkout -b feature
  Switched to a new branch 'feature'

- git branch
  *feature
  main

- git checkout main
  Switched to branch 'main'
  Your branch is up to date with 'origin/main'

- git branch
  feature
  *main

- git checkout -b testbranch
Switched to a new branch 'testbranch'

- git branch -d testbranch
error: Can not delete branch test branch' checked out at 'c:/Users/HP/Desktop/demo/test2'

- git checkout main
Switched to branch 'main'

- git branch -d testbranch
Deleted branch testbranch (was dc32653)

- git branch
feature
* main

- git push --set-upstream origin feature ← To push to feature branch

→ git diff branchName = To compare commits, branches, files & more

→ git ~~merge~~ merge branchName = To merge 2 branches

e.g. To compare 'feature' branch to 'main'
git diff main

→ PR (Pull Request) (On github online)
As soon as we push into feature branch, In this branch, there will be a green button named "Compare & Pull Request", click on it

→ Now, review it (You can comment on it)
→ Merge it.
→ commit it.

* Pull

→ We have merged feature to main online but vscode doesn't know that

→ git pull origin main ⇒ Used to fetch and download content from a remote repo and immediately update the local repo to match that content

* Merge Conflicts

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

→ If one branch made changes at the same place second branch made another change then git won't know which one it should remove.

- git branch
  feature
  *main

→ add in app.js file in first line button

- git add .
- git commit -m "button added"
- git checkout feature
- git branch
* feature
  main

→ Replace "button" with "form" in app.js
- git add .
- git commit -m "form added"

- git diff main
- git checkout main
- git diff feature
- git merge feature
  Auto-merging app.js
  CONFLICT (content): Merge conflict in app.js
  Automatic merge failed: fix conflicts and then
  commit the result

→ In VS Code editor window: There will be options on the TOP:
  Accept Current Change (HEAD)
  Accept Incoming change (feature)
  Accept both changes
  Compare changes
  (we will not use them)
→ Now, delete extra lines, we will stay with both change.
- git add .
- git commit -m "merge with feature"
- git push origin main
  verify in github online

* Fixing Mistakes

Case 1 : Staged changes (Undo add)

git reset <file name>

git reset

Case 2 : Committed changes (for one commit)

(Undo add & commit)

git reset HEAD~1

Case 3 : Committed changes (for many commits)

git reset <commit hash> ⇒ but changes will be highlighted

git reset --hard <commit hash>

↳ changes will be removed

→ git log (q for quit)

In this output - There are logs of our commits

Commit 1d21cf..... (HEAD → main, origin/main)

↑ commit hash

Merge : ...

Author : ...

Date : ...

message

Forking

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

→ Fork is rough copy.

→ Suppose, A person has github account named nodejs and has public repo named node.

→ We can not make changes in codes of another person's github repo. So, click on fork, a window for create new fork will open.

→ Give a name of repo, it will become our repo in our account

→ make some changes, save it, then click on create a pull request

→ Now, On nodejs's node repo there will be our pull request on their tab, they can review it.