Date: M T W T F S S

1). GIT ⇒ Version Control System is a tool that helps to track changes in code.

. GIT is a version Contod system.

* Track the history
* Collaborate.

2). GITHUB ⇒ website that allows developers to store & manage their code using Git.

* Repository → Project.

GIT Commands

git --version. → check git version

* Important note:
Configuring Git:

git config --global user.name "____"
git config --global user.email "____"
git config --list.

Date:

Clone & Status.

clone:→ cloning a repo. on our local machine.

git clone <--some link-->

status:→ displays the state of the code.

git status.

ls → list files

ls -a → list hidden files.

* Untracked file

new files that git doesn't yet track

(commit) * Modified changed

Important note:

(add) * staged file is ~~already~~ ready to be committed

* unmodified unchanged.

Date: M T W T F S S

add - adds new or changed files in your working directory to the git staging area

git add <-file name->

commit - It is the record of change

git commit -m "some message".

Push - upload local repo content to remote repo.

git push origin main.

Important note:

Init Command

DAY 2

init - used to create a new git repo

git init
git remote add origin <--link-->

git remote -v (to verify remote)
git branch (to check branch)
git branch -M main (to rename branch)
git push origin main.

mkdir localrepo
cd localrepo
ls -a (no .git)
git init
ls -a (exists .git)

Important note: * create new file
↳ add content.

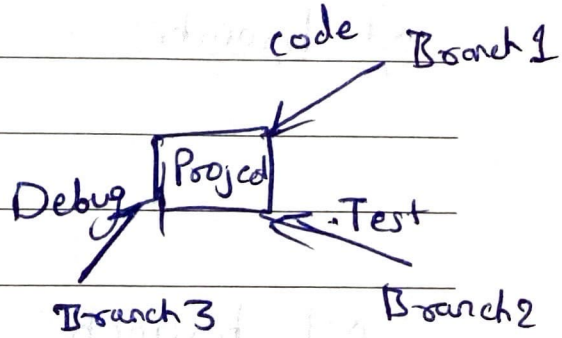
git status
git add .
git status

git commit -m "Message"
git status.

* create new repo on github. (without Readme)

git remote add origin <-- link of repo -->

git remote -v



git branch.

git branch -M main

git branch

git push origin main.

Important note:

git push -u origin main

↳ to set upstream.

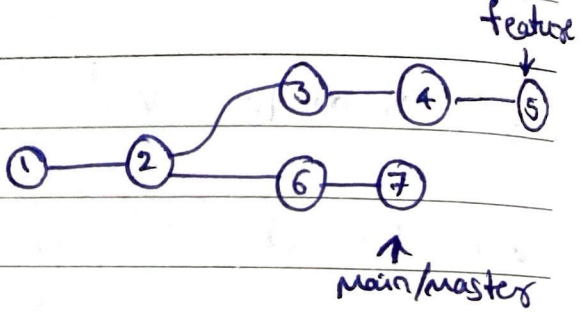
Date: M T W T F S S

create new file

git add .
git commit -m "Msg"
git push.

* Work Flow :→

git branches



git branch (to check branch)
git branch -M main (to rename branch)
git checkout <- branch name -> . (to navigate)
git checkout -b <- new branch name -> (to create new branch)

Important note:

git branch -d <- branch name -> (to delete branch)

Date: M T W T F S S

```
git branch
git branch checkout -b featurex → new branch
git branch
git checkout main
git branch
git checkout featurex
git branch
git checkout -b featurey
git branch
```

```
git branch -d featurey    (gives error)

git checkout main
git branch -d featurey    (deletes ✓)
git push origin featurex  (to gitHub)
```

Important note: git diff main

Merge

- git merge main. — Way 1
- Pull Request — Way 2.

Date: M T W T F S S

git pull origin main

[from GitHub (Remote) to Git(Local)]

* Resolving Merge Conflicts

An event that takes place when Git is unable to automatically resolve differences in code b/w 2 commits.

* When 2 same files in 2 different branches have changes in a same line then choose 1 or both if required then save the file.

git status.
git add.
git commit -m "____"

Important note:

git status.
git diff
git diff main
git checkout main
git merge featurex. ⇒ git push

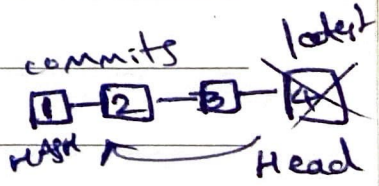
Date: M T W T F S S

* Undoing changes (added but not committed)

case 1 : staged changes.

git reset <--file name-->
git reset.

case 2 : commited changes (for one commit)



git reset HEAD ~1.
git log. (commits)

case 3 : commited changes (for many commits)

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

* FORK

Important note:

A fork is a new repo. that shares code & visibility settings with the original "ustream" repo.

fork is a rough copy.