GIT AND GITHUB

Date
Page

* git and github are different from each other.
  ↓ ↓
  Software Service

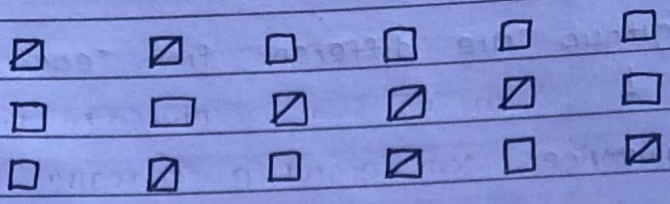
* Git is a version control system that allows you to track changes to your files and collaborate with others.
  It is just same as checkpoints in a game.
  It is free and open source software that is available for windows, macOS and linux.

* Github is a web based hosting service for git repositories.
  Github is an online platform that allows you to store and share code with others.
  Github is not only provider of git repositories : 1> Gitlab 2> Bitbucket 3> Sourceforge 4> AWS codecommit

* Before git there was SCCS (Source code control system) but it was expensive and not very user friendly.
  Some common version control systems are Subversion (SVN), CVS concurrent version system and perforce.

* Checkout git version
  git --version
  This command will display the version of git installed on your system.
  It is a stable software & no break out changes.

* Repository
  A Repository is like a folder on your computer, but it is more than just a folder. It can contain other files, folder and even other repositories.

* 

☑ → Tracked

☐ → Not Tracked

At any point you can run the following command to see the current state of your repository.

git status

* git init command will create a new folder on your system and initialize it as git repository.

This adds a hidden .git folder to your project.

* When you use ls -a command, you will see all the hidden folders.

. → Pointing to the current directory

.. → Pointing to the parent directory

.git → git repository

*

.git

1> HEAD

2> CONFIG

3> DESCRIPTION

4> HOOKS

5> INFO

6> OBJECTS

7> REFS

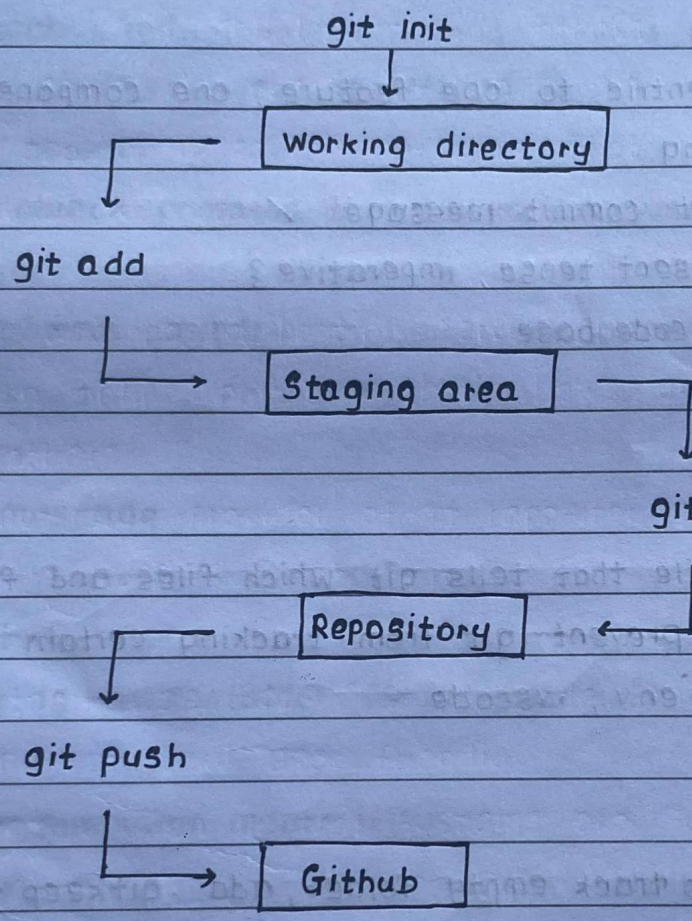
{ vi → visual interface }

Date
Page

* Commit

Commit is a way to save your changes to your repository. You can think of a commit as a snapshot of your code at a particular point in time.

*



* Stage

git add . || git add <filename>

After doing git add, our files are in staging area, this means that we have not yet committed the changes but are ready to be committed.

* Commit

git commit -m "your file is added"

{ message }

To comeback to first commit,
{ git reset --hard commit_id }

* Logs

git log || git log --oneline

This command will show you the history of your repository.

You can use the --oneline flag to show only commit message.

* Atomic commits

keep commits centric to one feature, one component or on fix.

Focus on one thing.

→ Present or past commit message ✓

Depends { Present tense, imperative }

give order to code base

Don't care

* gitignore

Gitignore is a file that tells git which files and folders to ignore.

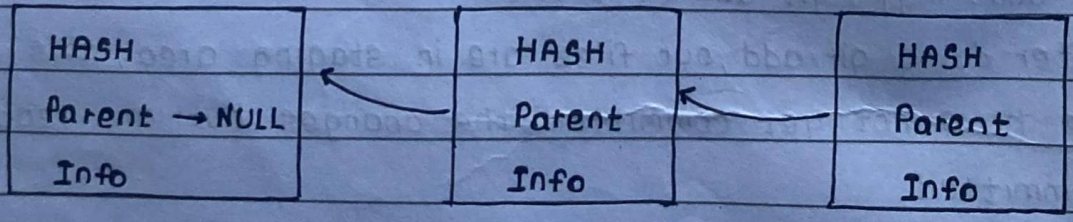
It is a way to prevent git from tracking certain files or folders.

node_modules, .env, .vscode

* gitkeep

When you have to track empty folder, add .gitkeep file in it.

* Commit behind the scene



* 3 Musketeers of git

1> Commit Object

2> Tree Object

3> Blob Object

* Commit Object

Each commit in the project is stored in .git folder in the form of a commit object.

A commit object contains the following information.

1> Tree Object

2> Parent commit Object

3> Author

4> Committer

5> Commit message

* Tree Object

Tree object is a container for all the files and folders in the project.

It contains following information.

1> File mode

2> File name

3> File hash

4> Parent Tree object.

5> Blob object

* Blob Object

Blob object is present in the tree object and contains actual file content.

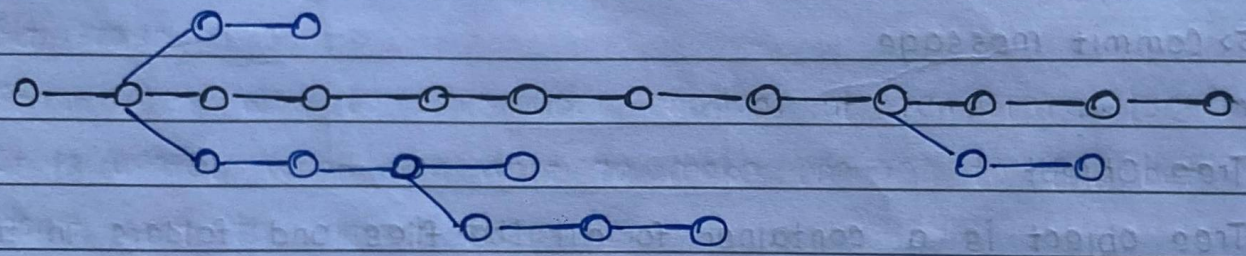
* There are 2 types of commands

1> Gardening commands : Low level commands
make SHA (Secure Hash Algorithm)
Add Indexing , then manually commit.

2> Porcelain commands : High level commands designed for human interaction and ease of use.

* Branches in git

Branches are a way to work on different versions of a projects at the same time. They allow you to create a seperate line of development that can be worked on independantly of the main branch.



* Head

The head is a pointer to the current branch that you are working on. It points to the latest commit in that current branch.

* git branch

This command lists all the branches in the current repository.

* git branch bug-fix

This command creates a new branch called bug-fix.

* git switch bug-fix || git checkout bug-fix

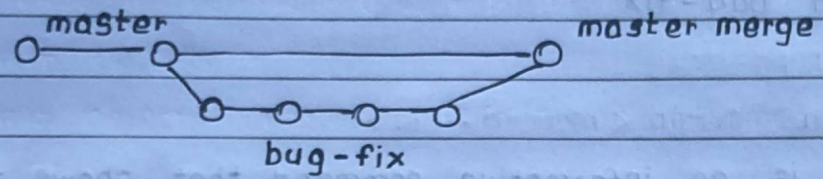
This command switches to the bug-fix branch.

* git switch -c dark-mode

This commad creates a new branch called dark-mode. The -c flag use to create new branch.

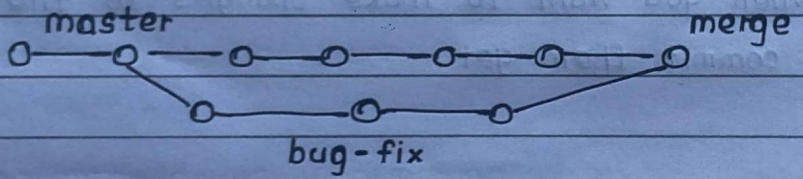
* Merging branches

1> Fast forward merge



This is fast forward merge. It means that the commits in the bug-fix branch are directly merged into the main branch. You are not doing anything on master branch.

2> Not fast forward merge



In this type of merge, the master branch also worked and have some commits that are not in bug-fix branch. This is a not fast-forward merge.

Command : On master branch,

git merge bug-fix

The main difference between fast forward and not fast forward merge is resolving the conflicts.

In a fast-forward merge, there are no conflicts.

In a not fast-forward merge, there are conflicts, and there are no shortcut to resolve them, you have to do it manually.

* Rename a branch

You can rename a branch using the following command.

git branch -m <old name> <new name>

* Delete a branch

You can delete a branch using the following command.

git branch -d bug-fix

* Git diff

The git diff is an informative command that shows the differences between 2 commits.

git diff <Commit-hash-one>..<Commit-hash-two>

* Git stash

Stash is a way to save your changes at temporary location. It is useful when you want to make changes to file but don't want to commit them yet.

git stash

View the stash list

git stash list

Apply the stash : Taking out of stash

git stash apply

Drop the stash

git stash drop

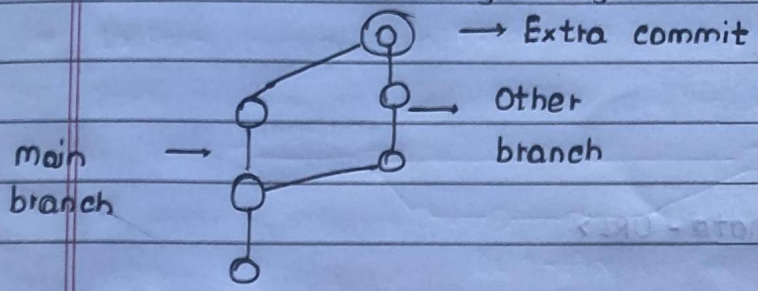
* Git tags

Tags are a way to mark a specific point in your repository. Generally, there are used by production managers.

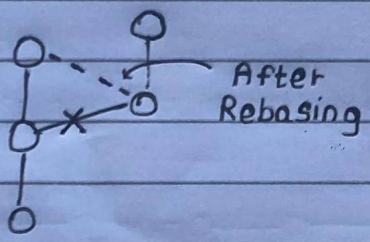
* Rebase

This command is used to avoid extra commit when we merge main branch into other branch.

When we do it using merge



When we do it using Rebase



Command to run while working in other branch,

git rebase master

* Reflog

Git reflog is a command that shows you that history of your commits.

git reflog

Date
Page

* Getting started with github

1 Setup SSH key & add it to github.

2 Check remote URL setting

git remote -v

3 Add remote repository

git remote add origin <remote-URL>

4 Push your code to remote repository

git push origin main