Git and GitHub

Kafui

Git,

Can be liken to as a Software program, that allows us to keep track of changes in our projects over time.

How it works?

- It records changes in our projects
- stores the changes and
- reference them as and when needed

Basic Git Workflows

git init

Before we start tracking changes in our projects, we need to let Git know about it - using the command:

*__git init__,

- helps us to initialize the beginning of our project.

The Workflow

Once we have a Git project, our projecthas a three part structure:

01.

A working directory:

best known to be our local environment (pc) where we write, edit, delete and organize files.

02.

A Staging area:

Where we listall of our changes we made to the working directory.

03.

A repository:

Where Git permanently stores the changes we made as different versions of our project.

Basic Git Workflows

git status

We need to keep track of changes we make to our project hence, we check the status of our working directory using the command:

*__git status__

- reveals what changes are currently on going in our project.

Basic Git Workflows

git add “filename”

Git at this point still hasnt start to track changes until we tell it to so. In other to do that we use the command:

*__git add “filename”__

- which adds our file(s) to the staging area.

Basic Git Workflows

git diff “filename”

When we want to check for differences in our project - changes that have been made and not traacked, we use the command:

*__git diff “filename”__

- which checks our file(s) for differences in changes.

Basic Git Workflows

git commit -m “message”

Commits: are snapshots of a branch in a repository. A commit will permanently store our changes from staging area to the repository. we use the command:

*__git commit -m “message”__

- which checks our file(s) for differences in changes.

Basic Git Workflows

git log

Often times, we will need to review or go back in time to check our changes using the command:

*__git log__

- it display a 40 char SHA, unique identifier of our commit.
- commit authus: You
- date and time of commit
- commit message

Git Branching

git branch:

A **branch**:

is an independent version of the main project (codebase).

This helps to create multiple (duplicates) of the codebase to experiment with features.

***git branch**:

lists all the available branches in your project if there is with an * **(asterisk)** attached to the current one.

GitHub

Is a website/service that allow us to put our code or store them in containers known as repositories, tracking changes made to the code.

It as well offers hosting services and tools to **build**, **test**, and **deploy** code.

The difference?:
Git is a development tool used outside of GitHub,

whilst

GitHub is a service.

The GitHub Workflow

01

create a branch

02

commit your changes

03

create a pull request

04

review a pull request

05

merge and
delete branch

Code Contribution Landscape

Whenever we're working on a project code, each teammate will have to own a copy of the main code repository that he/she can add a feature, fix a bug or recommend changes to. Doing so will mean creating a branch off of the main project code so the original code doesn't get interfered with and only accessed when the new code is tried and tested before merging into the main code.

Say you have been added to a team to contribute code to an ongoing project, after having cloned (make copy) of the main project from GitHub to yoour local environment, your careted a new branch to add a feature, you then have to commit those new feature codes you added to record your changes. After which you push the new commit to the GitHub repository.

Now your work is ready to be reviewed by the maintainers of the project before it can be integrated into the main project.

Once a pull request is created, other members on the team, can review and possibly merge your pull request into the main project.

Now that we have the pull request approved and merged eventually, we need to keep files and things organized, hence it will be ideal to delete closed works (pull requestr) and leave those that are still opened - in progressing.

Final reflections and future steps

Practice!, practice!!, practice!!!

Thank you!