GIT & GitHub? What are they?

Every single time I heard about Git and GitHub, i thought they were the same, until yesterday. Thanks to my kind seniors!

Before going in let's see version control system. Version control system is basically keeping track of all the versions of a project or a code which you have done until then. So if you have VCS you can access any of your past versions instantly. Likewise, for a group project it is not just about you but each members are assigned different parts so that you do your part most effectively. In such a project we use distributed VCS. And that's where GIT comes in.

Git is a distributed version control system, which means that a local clone of the project is a complete version control repository. These fully functional local repositories make it easy to work offline or remotely. Each members can have a local repository/folder where they can edit and do their work as they want without affecting others work. Once your work is ready you can sync your copy of their folder with the group.

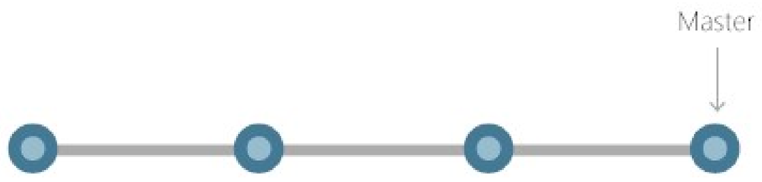
Git's flexibility and popularity make it a great choice for any team. Many developers and students like us already know how to use Git. It does matter that we learn it too!

Git's user community has created resources to train developers and Git's popularity make it easy to get help when needed. Nearly every development environment has Git support and Git command line tools implemented on every major operating system.

How to use Git?

Most times GIT is run using GIT Bash.

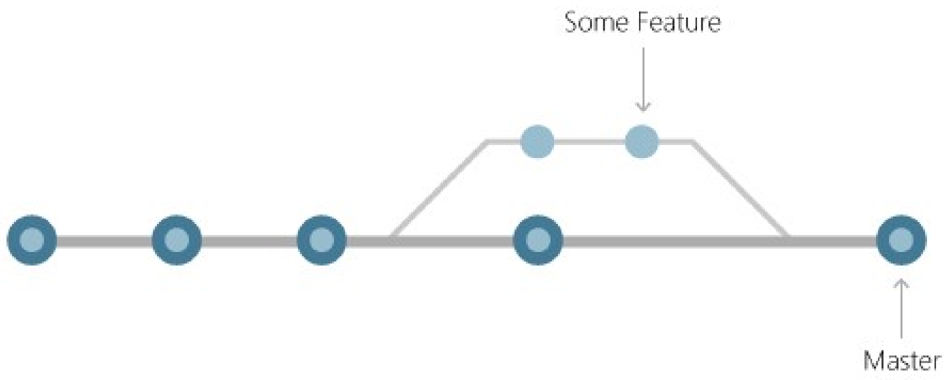
Every time work is saved, Git creates a commit. A commit is a snapshot of all files at a point in time. If a file hasn't changed from one commit to the next, Git uses the previously stored file. This design differs from other systems that store an initial version of a file and keep a record of changes over time.



Commits create links to other commits, forming a graph of the development history. It's possible to revert code to a previous commit, inspect how files changed from one commit to the next, and review information such as where and when changes were made.

Branches:

Each member saves changes to their own local folder/repository. As a result, there can be many different changes based off the same commit. Git provides tools for isolating changes and later merging them back together. Branches, which are lightweight pointers to work in progress, manage this separation. Once work created in a branch is finished, it can be merged back into the team's main (or trunk) branch.



Files and Commit:

There are three major stages for a file during development:

1. Modified: First you create and modify your file in the local folder. Technically file development
2. Staged : To include your file in the main/Master branch after you commit you need to stage it. Basically rehersal.
3. Committed : Only a staged file is committed which then includes it in main/Master link. That is The Final Show