GIT ref: <https://hackernoon.com/understanding-git-fcffd87c15a3>

\* [GitHub](https://hackernoon.com/tagged/github) is *not* git

\* '**git**' is British slang for a dumb, annoying, or generally unpleasant person.

\* Linus Torvalds Invented **Git**, But He Pulls No Patches With GitHub. Linus Torvalds keeps a copy of his Linux kernel project on GitHub, the wildly popular code-hosting website

**What is** [GitHub](https://github.com/) ?

[GitHub](https://github.com/) is a website for hosting projects that *use* git.

**GitHub is** an open-source repository hosting service, sort of like a cloud for code. It hosts your source code projects in a variety of different programming languages and keeps track of the various changes made to every iteration.

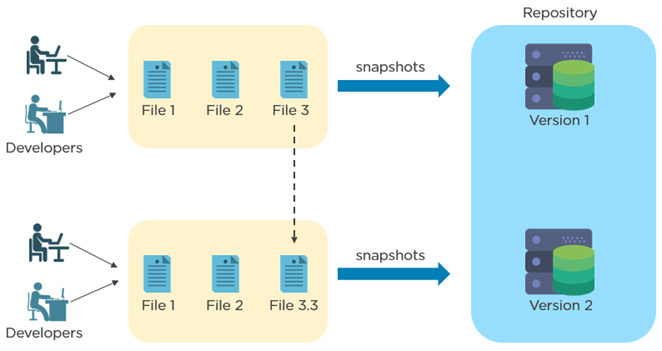
**What is git?**

\* **Git** is a **Distributed** **Version Control tool** that is used to store different versions of a file in a remote or local repository. It is used to **track changes in the source code**.

\* For example, when you edit a file, git can help you determine exactly what changed, who changed it, and why.

\* It allows multiple developers to work together. A VCS allows you to keep every change you make in the code repository. VCS stores changes as **Versions** by taking a **snapshot** of every change.

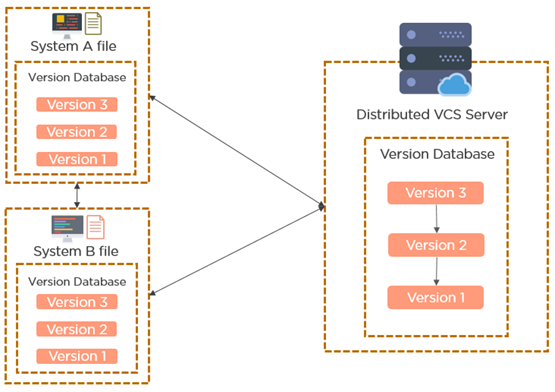
Here a how the diagram of a VCS looks like:



\* As you can see, Developers are making changes to the file and each version of the file is being stored in a repository. This repository is called **GitHub** repository.

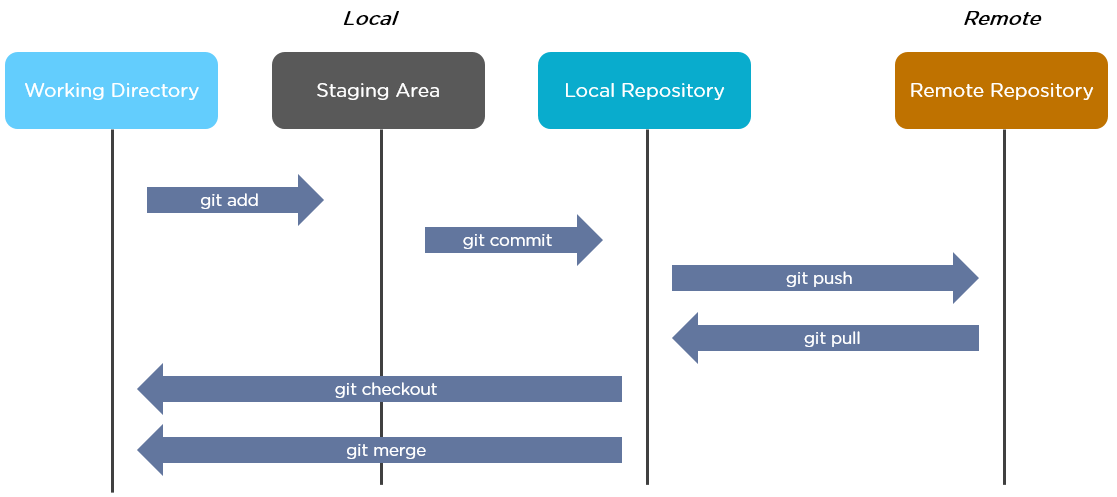
\* Git is a **Distributed Version Control** tool that allows multiple users/developers to write codes, edit and make commits to the server repository (GitHub).

Here is a diagram that shows how a distributed VCS architecture looks like:



\* As you can see, every developer has a copy of the entire code along with its history. This allows all the developers to update their local repositories with new data from the server (GitHub).

The basic architecture of Git can be shown as:



Let me explain about the architecture:

* Working directory is your current location of the folder you are working with.
* Then you add your files to the staging area before saving the changes (commit)
* After all the changes are made, you commit the files to the local repository
* Then you push the committed files to the remote repository
* If needed, you can the push the changes from the remote repository to the local repository
* Git allows you to create branches and switch to them when required
* After the changes are complete, you can merge the newly created branches to the master branch

ref:

1) branching <https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell>