**Links: https://www.educba.com/introduction-to-git/**

**Git:** (global information tracker):

* A free and open source software for distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
* Git is used to track changes in any set of files
* During software development , git is used for coordinating work among programmers collaboratively developing source code.

**GitHub:**

* Github is a code hosting platform for version control and real-time collaboration, it lets you and others work together on projects from anywhere.
* It is a web-based interface that uses git, allows multiple people to make separate changes to web pages at the same time.

**Difference b/w Git & GitHub:**

* Git is a tool that’s used to manage multiple versions of source code edits that are then transferred to files in a git repository, GitHub serves as a location for uploading copies of a Git repository.

| **s.no** | **Git** | **GitHub** |
| --- | --- | --- |
|  |  |  |
| **1** | Git is a software | GitHub is a service |
| **2** | Git is a command-line tool | It is a Graphical User Interface(GUI) |
| **3** | Git is installed locally on the system | GitHub is hosted on the web |
| **4** | Git is maintained by linux and created by Linus Torvalds | Maintained by Microsoft and created by Tom preston |
| **5** | It is focused on version control and code sharing | It is focused on centralized source code hosting. |
| **6** | Version control system to manage source code history | It is a hosting service for Git repositories |
| **7** | First released in 2005 | Launched in 2008 |
| **8** | Has no user management feature | It has a built-in user management feature |
| **9** | Open-source licensed | It has a free-tier and pay-for-use tier. |
| **10** | It has minimal external tool configuration | It has an active marketplace for tool integration. |
| **11** | It provides a Desktop interface named Git GUI | It provides a Desktop interface named GitHub desktop |
| **12** | It competes with CVS,Azure DevOps server, Subversion, Mercurial, etc. | It competes with GitLab, Git Bucket, AWS Code Commit,etc. |

**Version control system:**

* Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.
* A version control system is a kind of software that helps the developer team to efficiently communicate and manage(track) all the changes that have been made to the source code along with the information like who made and what changes have been made.

**Types of Version Control Systems:**

* Local Version Control Systems
* Centralized Version Control Systems
* Distributed Version Control Systems (Git)

## Centralized Version Control

Centralized Version Control is a version control system using server/client model and server contains all the history of source code. Here, the server acts as the main repository.

## Distributed version control

With distributed version control systems (DVCS), you don't rely on a central server to store all the versions of a project’s files. Instead, you clone a copy of a repository locally so that you have the full history of the project. Two common distributed version control systems are Git and Mercurial.

Git is a distributed version control system (DVCS). "Distributed" means that all developers within a team have a complete version of the project. A version control system is simply software that lets you effectively manage application versions. Thanks to Git, you'll be able to do the following:

* Keep track of all files in a project
* Record any changes to project files
* Restore previous versions of files
* Compare and analyze code
* Merge code from different computers and different team members

**Git repository** is just a file location where you are storing all the files related to your project.

Let’s say if you are developing an application, whatever you are coding, all the different service modules for your application will be dumped in your **Git repository** once you *commit* & *push*.

When you are working with Git, you have two repositories - **Local** & **Remote**.

* **Local repository**: It is just a file location residing in your system. When you ***git commit*** your code, a version/snapshot is created in your local repo.
* **Remote repository**: A remote repository generally lies somewhere outside your system, on a remote machine. This is very important when you are working with multiple people. This is the place where everyone will be sharing their code.

You can add files in your remote repo by ***git push*** from your local repository.

**Git Commands:**

* [**git ini**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-init)**t** creates a new Git repository
* [**git add**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-add)adds new or changed files in your working directory to the Git staging area.
* [**git commit**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-commit) **It is used to record the changes in the repository.**
* [**git status**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-status) **shows the current state of your Git working directory and staging area.**
* [**git remote**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-remote) **A remote in Git is a common repository that all team members use to exchange their changes, The git remote command lets you create, view, and delete connections to other repositories.**
* [**git push**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-push) **used to upload local repository content to a remote repository.**
* [**git clone**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-clone) **is used to make a local copy of a remote repository.**
* [**git branch**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-branch) **lets you create, list, rename, and delete branches.**
* [**git checkout**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-checkout) **lets you navigate between the branches created by git branch**
* [**git log**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-log) **a utility tool to review and read a history of everything that happens to a repository**
* [**git stash**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-stash) **takes your uncommitted changes (both staged and unstaged), saves them away for later use, and then reverts them from your working copy.**
* [**git revert**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-revert) **a forward-moving undo operation that offers a safe method of undoing changes.**
* [**git diff**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-diff) **to track the difference between the changes made on a file.**
* [**git merge**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-merge) **Git merge is a command that allows you to merge branches from Git.**
* [**git rebase**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-rebase) **the process of moving or combining a sequence of commits to a new base commit.**
* [**git fetch**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-fetch) **used to download commits, files and references from a remote repository into the local repository.**
* [**git reset**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-reset) **used to undo local changes to the state of a Git repo.**
* [**git pull**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-pull) **The git pull command fetches and downloads content from the remote repository and integrates changes into the local repository.**

**Shivakumar Ummeda**