**Git** **:** 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 a version control system used for tracking changes in computer files. It is generally used for source code management in software development.)

**Uses :**

* 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.
* It supports non-linear development through its thousands of parallel branches.

**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 systems(VCS):**

* 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 System:** 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.

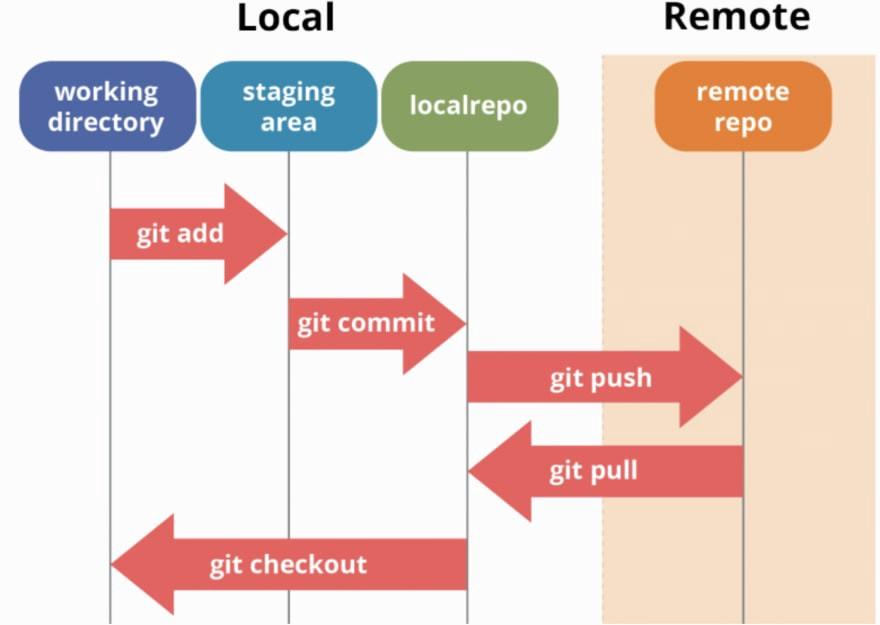
* Uses a central server to store all the files
* Every operation is performed directly on the repository
* All the versions of the file are stored on the Central VCS server
* In case the central server crashes, the entire data of the project will be lost. Hence, distributed VCS was introduced.

## **Distributed Version Control System:** 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.

* Every [programmer](https://www.simplilearn.com/how-to-become-programmer-article) has a copy of all the versions of the code on their local systems
* Record any changes to project files, they can restore previous versions of files
* They can update their local repositories with new data from the central server and changes are reflected in the principal repository

# **3 main parts of git:**

1. **Working directory** – This is the directory where all your project files and folders reside (along with the .git folder). Each of your files within this directory is in 1 of possible states, untracked, unmodified, modified, staged.
2. **staging area (index)** – This is a hypothetical layer which sits on top of the last commit’s layer. When you run the “git commit” command the 2 layers get merged. Any files that are in the commit layer, that has a newer file directly above it (in the “staging layer”), will get over-written by the newer (staged) file. Note, that a file’s content is tracked so that you can roll back to how the file looked like in any previous (commits) snapshots. Staging area is also referred to as the “index”
3. **git directory (repository)** – this is basically like your git repository database.



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

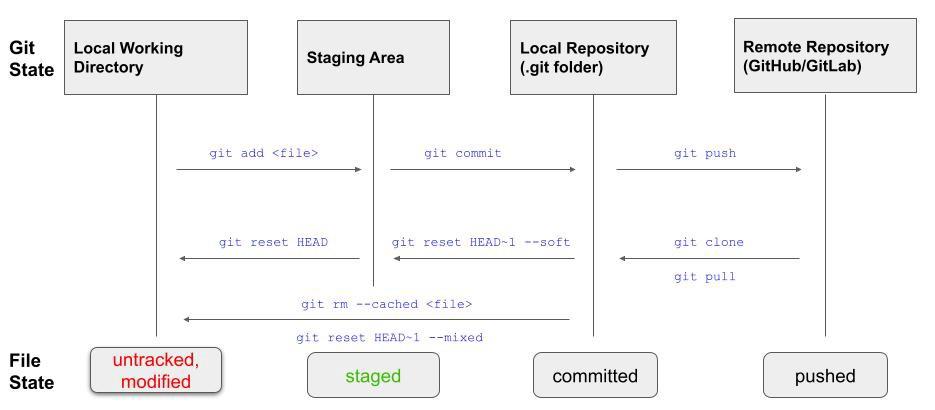
Let’s say if you are developing an application, whatever you are coding, all the different service modules for your application will be dumped in your Git repository once you commit & push.

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 commityour 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.

**Branch of a Repository:** A repository is your whole project (directories and files) that you clone on your computer. A branch is a version of your repository, or in other words, an independent line of development. A repository can contain multiple branches, which means there are multiple versions of the repository.

## **Different Git states and corresponding file states:**

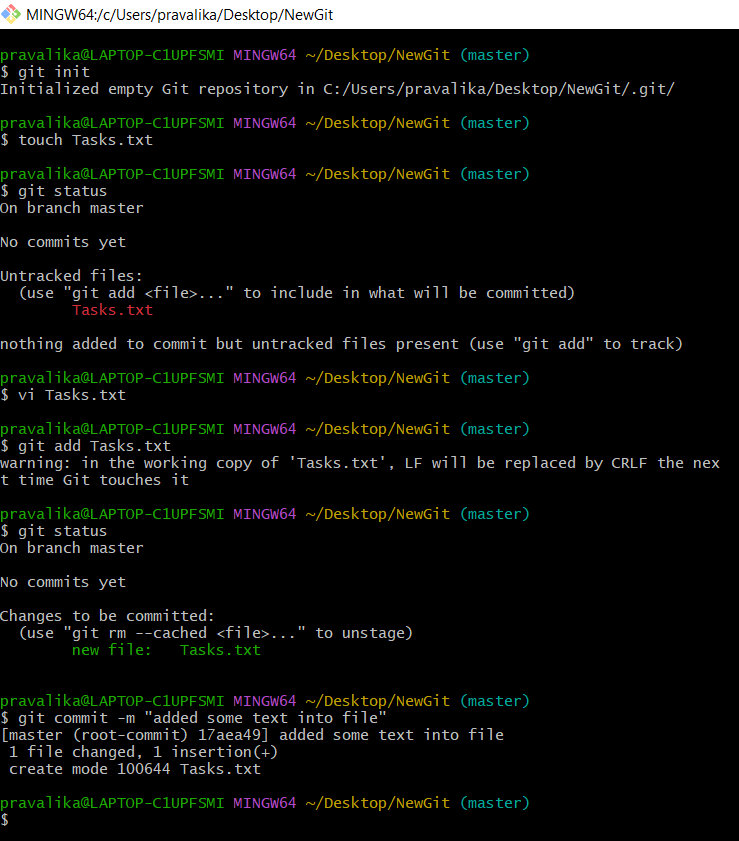


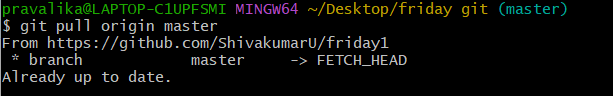
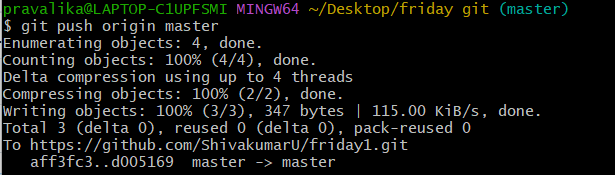
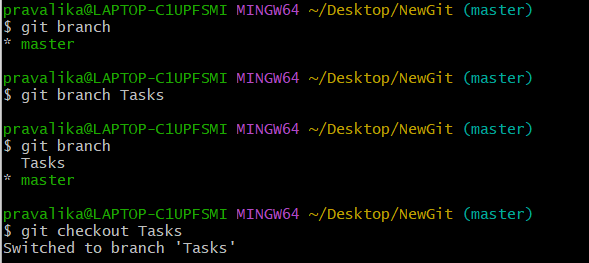
**Advantages of Git:**

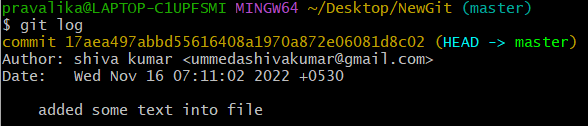
* Good distributed model as each developer gets a local repository with a full history of commits which makes git fast compared to other VCs.
* Branching capabilities and merging are easy (as they are cheap), good data integrity
* The push/pull operations are faster as they save time and developers can fetch and create pull requests without switching.

**Git COMMANDS :**

* **G**[**ini**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-init)**t** creates a new Git repository
* **touch** creates a new file
* **Vi** vi stands for visual instrument. It is a widely-used default text editor for Unix-based systems
* **Git add (**adds new files or changed files in your working directory to the Git staging area)
* **G**[**it commit**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-commit) records the changes in the repository.
* **G**[**it status**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-status)shows the current state of your Git working directory and staging area.
* **G**[**it 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 pull** The git pull command fetches and downloads content from the remote repository and integrates changes into the local repository.
* **G**[**it push**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-push)used to upload local repository content to a remote repository.
* **G**[**it clone**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-clone) is used to make a local copy of a remote repository.
* **G**[**it branch**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-branch)lets you create, list, rename, and delete branches.
* **G**[**it checkout**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-checkout) lets you navigate between the branches created by git branch
* **G**[**it 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



* **G**[**it 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.
* **G**[**it 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.
* **G**[**it diff**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-diff) to track the difference between the changes made on a file.
* **G**[**it 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.
* **G**[**it 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.
* **G**[**it 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.
* **G**[**it reset**](https://intellipaat.com/blog/tutorial/devops-tutorial/git-commands/#git-reset)used to undo local changes to the state of a Git repo.