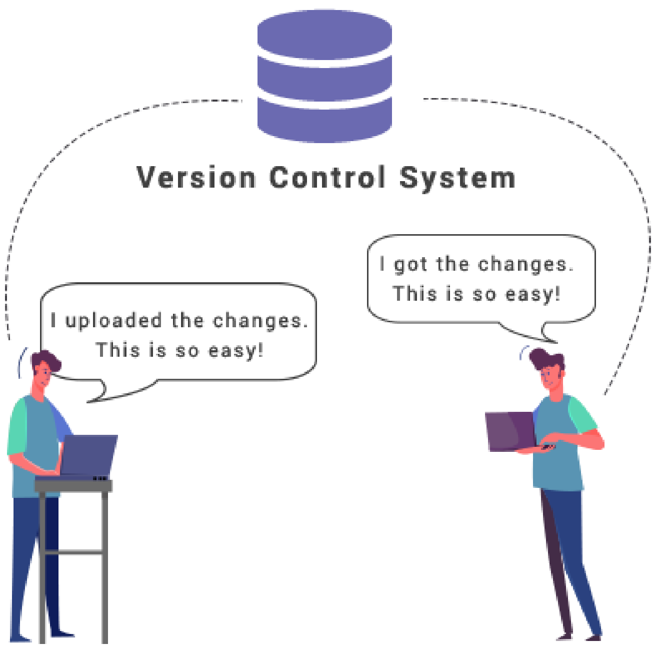
Chapter 1

GIT as Distributed Version Control system

1.1.GIT

GIT is the most popular Version Control System nowadays because it is an open-source software which is easy to handle and perform work on various projects. GIT Version Control System (GIT VERSION CONTROL SYSTEM) is a system that manages the development of an evolving object. In other words, it is a system that records any changes made by the software developers. There are a lot of uses for GIT VERSION CONTROL SYSTEM in software development that makes the development process easier and faster. GIT VERSION CONTROL SYSTEM is also known as Revision Control System.

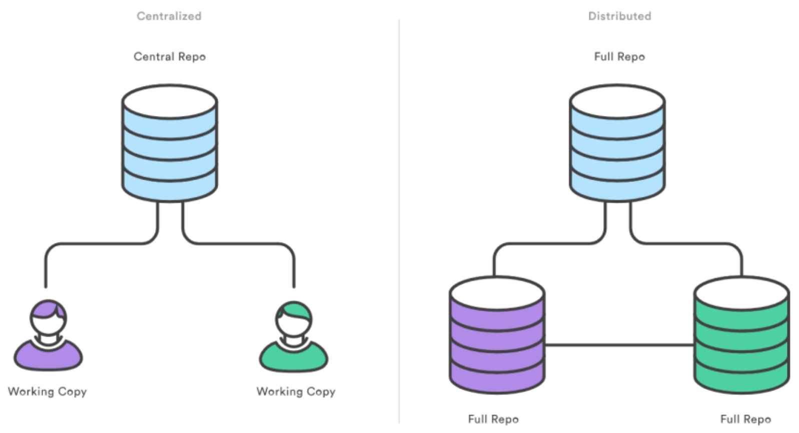

Version Control System
I uploaded the changes. This is so easy!
I got the changes. This is so easy!

In the software development process, it is normal for software developers to continually make changes in pieces of codes and other files that involve addition and deletion of a feature. It is realized that several revisions will be made before producing the final version. It is difficult to manage and organize the codes and the files because the number of revisions grows larger due to larger and complex systems. Hence, the existence of the GIT VERSION CONTROL SYSTEM really helps software developers to accelerate and simplify the development process. Without GIT VERSION CONTROL SYSTEM, the software developers are enticed to keep various duplicates of code on their computer. This is risky because it is easy to change or erase a document or file in the wrong copy of code, possibly leading to losing work. Version Control System will take care of this issue by managing all versions of the code. The adoption of GIT VERSION CONTROL SYSTEM is progressively mandated to empower all the software developers who work on the same project to work together effectively towards project milestones. It is an approach of managing the tasks of multiple team members who are sometimes at different locations.

1.2.Features of GIT

Here are some basic and most important features of GIT:

a. **Distributed System:**
In SVN, each developer gets a working copy that points back to a single central repository. Git, however, is a distributed version control system. Instead of a working copy, each developer gets their own local repository, complete with a full history of commits.



Having a full local history makes GIT fast, since it means you don't need a network connection to create commits, inspect previous versions of a file, or perform diffs between commits. Distributed development also makes it easier to scale your engineering team. If someone breaks the production branch in SVN, other developers can't check in their changes until it's fixed. With Git, this kind of blocking doesn't exist. Everybody can continue going about their business in their own local repositories. And, similar to feature branches, distributed development creates a more reliable environment. Even if a developer obliterates their own repository, they can simply clone someone else's and start new work.

b. **Compatibility**
GIT is compatible with all the Operating Systems that are being used these days. GIT repositories can also access the repositories of other Version Control Systems like SVN, CVK, etc. GIT can directly access the remote repositories created by these SVNs.
So, the users who were not using GIT in the first place can also switch to GIT without going through the process of copying their files from the repositories of other GIT VERSION CONTROL SYSTEM's into GIT-GIT VERSION CONTROL SYSTEM. GIT can also access the central repositories of other GIT VERSION CONTROL SYSTEMs.

c. **Non-linear Development:**
GIT allows users from all over the world to perform operations on a project remotely. A user can pick up any part of the project and do the required operation and then further update the project. This can be done by the Non-linear development behavior of the GIT. GIT supports rapid branching and merging and includes specific tools for visualizing and navigating a non-linear development history.

A major assumption in GIT is that a change will be merged more often than it is written. GIT records the current state of the project in a Tree form. A new branch can be added to the tree anytime and is further merged with the final project once it's complete.

d. **Easy Branching**
GIT allows its users to work on a line that runs parallel to the main project files. These lines are called branches. Branches in GIT provide a feature to make changes in the project without affecting the original version. The master branch of a version will always contain the production quality code. Any new feature can be tested and worked upon on the branches and further, it can be merged with the master branch.

Branch management with Git is very simple. It takes only few seconds to create, delete, and merge branches. Feature branches provide an isolated environment for every change to your codebase. When a developer wants to start working on something, no matter how big or small, they create a new branch. This ensures that the master branch always contains production-quality code.

e. **Lightweight**
GIT stores all the data from the central repository on to the local repository while cloning is done. There might be hundreds of users working on the same project and hence the data in the central repository might be very huge. One might be worried that cloning that much data into local machines might result in system failure but GIT has already taken care of such a problem. GIT follows the criteria of lossless compression that compresses the data and stores it in the local repository occupying very minimal space.
Whenever there is a need for this data, it follows the reverse technique and saves a lot of memory space.

f. **Speed**

Since you do not have to connect to any network for performing all operations, it completes all the tasks really fast. Performance tests done by Mozilla showed it was an order of magnitude faster than other version control systems. Fetching version history from a locally stored repository can be one hundred times faster than fetching it from the remote server.
The core part of Git is written in C, which avoids runtime overheads associated with other high level languages.

g. **Secure**

GIT keeps a record of all the commits done by each of the collaborators on the local copy of the developer. A log file is maintained and is pushed to the central repository each time the push operation is performed. So, if a problem arises then it can be easily tracked and handled by the developer. GIT uses SHA1 to store all the records in the form of objects in the Hash. Each object collaborates with each other with the use of these Hash keys.
SHA1 is a cryptographic algorithm that converts the commit object into a 14-diGIT Hex code. It helps to store the record of all the commits done by each of the developers. Hence, easily diagnosable that which commit has resulted in the failure of the work.

h. Reliable

Providing a central repository that is being cloned each time a User performs the Pull operation, the data of the central repository is always being backed up in every collaborator's local repository.
Hence, in the event of crashing of the central server, the data will never be lost as it can be gained back easily by any of the developer's local machine. Once the Central Server is all repaired, the data can be regained by any of the multiple collaborators.

i. Economical

GIT is released under the General Public's License(GPL) and hence it is available for free. GIT creates a clone of the central repository on the local machine and hence, all the operations are performed on the local machine of the developer before pushing it on to the central repository. Pushing is done only after the version on the local machine is working perfectly and is ready to be pushed on the central server. No experimenting is done with the files on the central server.

https://www.w3schools.com/git/
https://www.atlassian.com/git/tutorials

S. No.	GIT	GITHUB
1.	Git is a software.	GitHub is a service.
2.	Git is a command line tool.	GitHub is a graphical user interface.
3.	Git is maintained by linux.	GitHub is maintained by Mircrosoft.
4.	Git focused on version control and code sharing	GitHub focused on centralized code hosting