SNSE 375 Github Report:
Git VCS

Li Pan

Git is an open source version control system which robust and reliable. It is widely used today for tracking and managing the source code changes in software development.

1. Git History

Git development environment started out in Linux kernel. In the early 2000s, there are BitKeeper and Monotone which were both distributed version control systems made to manage the codebase in the Linux kernel for free. After both maker stop to cooperate with Linux kernel, Linux Torvalds has to develop scripts to manage email patches. As the scripts developed, they can be quickly merged so that the designer can keep modifying the codebase. After this, Git's design idea was formed.

Git's initial design goals were to achieve a similar function as BitKeeper that supports the distribution of versions and tracks the workflow in the security condition. Moreover, Git distributes architecture that supports collaborators can work on a diverse number of branches, each with their own steps. Git keeps track of all changes made and arranges merge work. Almost at the same time as Git was being developed, other open resource VCSs begin to develop. Mercurial is one VCS that is simpler than Git, and used in Facebook.

Today, the effectiveness of Git is that many open sourced version control system projects are in existence, such as Fossil and Bazaar. However, Git is still the most popular one.

2. Version Control

Having access to a version control system is very helpful in software development. It allows software developers to work on the same projects separately. It tracks the changes on every modification. Similarly, It supports the backup of older versions of the source code.

Most version control systems can restore the content and track the changes in the content. However, not all version control systems can keep track of the distribution with collaborators. Git contains all these feature as a version control system. It supports content storage, commit record, merge history, and manages different distributions.

Git uses directed acyclic graph (DAG) design to store content with different types of objects. Comparing the liner history the directed acyclic graph design is very powerful at content restorage. It can capture the different between each version and keep it as snapshot. It is easily restores any version repository in the history.

Commits and merge history can be approached with linear or directed acyclic graphic. Linear history only allows commits to be one after another without merged branches. Git uses directed acyclic graph to keep the change history, which allows for the unlimited use of

commits. Each commit contains the metadata and each commit can have many parents. Git also records all the merge history.

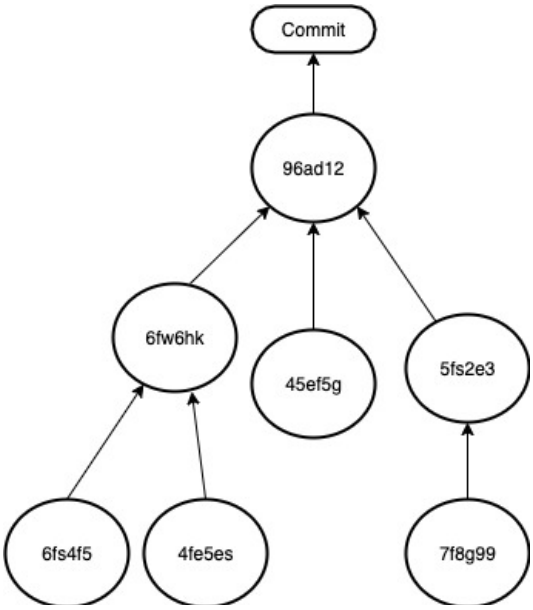

Figure 1 DAG of Git

All the commit node is directed to the root directory. Each commit can have multiple file changes, and the commit node can have multiple parents branches. When a directory node has different branches, it guarantees that two branches will have the same content. When we merge the different branches together, Git can compare the differences between content and keep changes together.

Generally, version control system can be localized, centralized, and distributed. Localized version control systems are simple and are only used in a local computer. However, it is hard to track which directory is currently being worked on. Thus, the localized version control system easily cause errors. Centralized version control system allows more than one developer to work on a project. The server only has one repository that includes all the versions. However, It has a serious downside; that is, if one developer commit to a change that causes the server to become corrupted, other developers are not able to keep working on their projects. The changes made by each developer on their own computer will be lost as well. Distributed version control systems overcome these drawbacks. In distributed version control, every developer has their own copy of repository which contains all the metadata. Each developer works on their own branch, even if their commit has errors that cause their copied repository to go down, it won't affect other developers' work. Git is a distributed version control system (See Figure2), and it has the advantages that allow collaborators to have a backup of their own repository version to catch up with the newest version on the repository server.

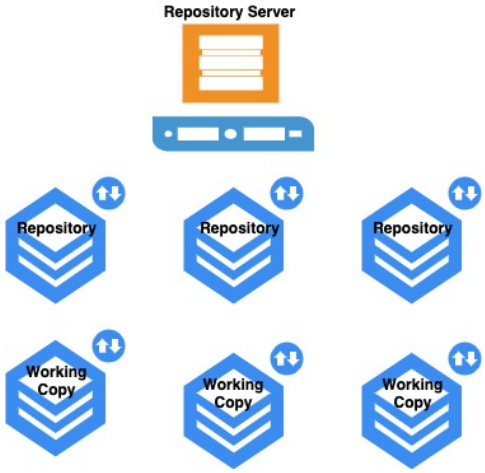

Figure 2 Distribute Version Control

3. Toolkit

Git was originally designed for Linux, and its toolkit includes many powerful command-lines. The low-level commands track and manage the content changes with directed acyclic graphs. The high level commands manipulate the different repositories. Many Graphical User Interface(GUI) tools are built on the top of the toolkit of Git.

4. Repository, index and working area.

A new repository can be created by running the git init command. It is easy to create a branch, commit, and tag with the repository. Besides, Git allows different local repositories to communicate with each other. Under git directory there are many different categories. Git configuration is the description of the local repository. It can be checked under git/config, git/description and git/info/exclude. Git hooks directory contains the scripts which run on the certain events of the repository. Git staging can check out under by git/index. Git object database is under the git/objects directory which contains all content or pointers to local content. Git references is for storing both local and remote branches references pointers. Git index is used for a staging area between working copy and local repository, and it lists the changes made on the staging. Using git status command can check the current index, and it indicates all the files that are modified on staging. In the Git work area developer can modify files. By using git add command it is possible to keep the changes in the staging area. The object ID is recorded as well.

5. The Object Database

The object database is set of directories In Git system. It allows developers to retrieve content anytime. Git contains four basic objects: tree, blob, commit, and tag. Each object has a type, size, and content. A tree is an object representing a content directory entries. A blob represents a file content under that repository. A commit contains the snapshot information and points to its top-level tree. A tag is a kind of label on a commit which contains the name in the repository history.

All objects are referenced by simple hashing algorithm(SHA) in Git. IF two objects are identical then they have the same SHA. Two properties of SHA points to the same object represent Git distributed model.

6. Storage and compression techniques

Git manages storage space by packing the objects in compressed format. Each index file corresponds to a pack file (See the Figure 4). Git efficiently manages delta compression at blob object level. Git makes it so that each commit only takes up a small amount of space. Git can calculate the commits and content that needs to be sent, then generate the pack file by using a client desired protocol for communication.

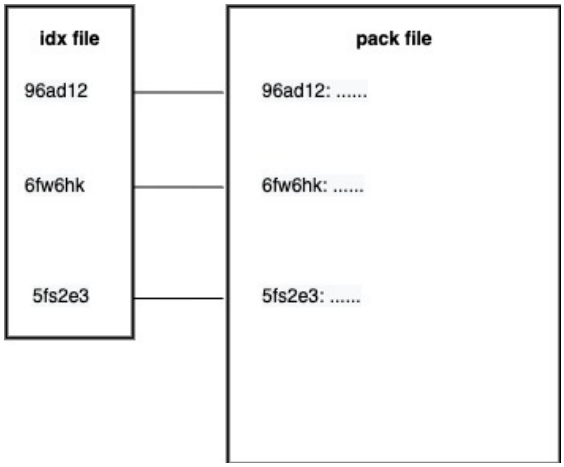

Figure 4 . Index file and Pack file

7. Merge history

Git can bring the complete changes to merge into to the master branch. The development history have two nodes join together into a new version. Each developer has a separate merge spot without affecting the work of others (See Figure 5).

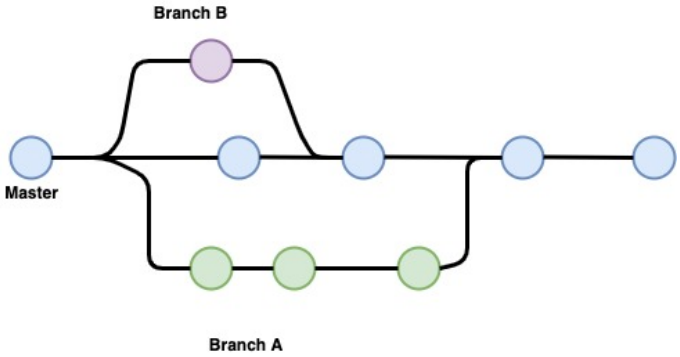

Figure 5. Git merge

Git supports a visualized merge history, it can be checked out by running the git log –graph command line.

Git supports many popular integrated development environments, and performs as a very powerful version control system. More and more programming platforms are becoming integrated with Git, such as NetBeans and Visual Studio Code. Ultimately, Git drastically helps software developers to improve their work efficiency.