Introduction to Git

life michael

Introduction

What is a Version Control System?

- Using a version control system we can track the history of our project files and revert that collection to another historic version at any point of time.
- There are three types of version control systems: local version control system, centralized version control system and distributed version control system.

Local Version Control Systems

- The local version control systems were the first to emerge. They allow us to maintain the history of our files on our local computer.

Centralized Version Control Systems

- The files are kept on a common accessible server that everyone can access from their local machines.
- Whenever a developer wants to edit a file only the last version of that file is retrieved.
- When a developer changes a file the change is automatically shared with all others.

Distributed Version Control Systems

- When using a centralized version control system we risk our files. There is the chance that we will lose the entire history of all files.
- Using a distributed version control system we enjoy both the benefits of using a local version control system and a centralized one.

Distributed Version Control Systems

- We can make changes on our local computer without worrying about the connectivity with the server.
- We don't rely on a single copy of all files stored on the server.
- Our files stored on the server are accessible to the other developers allowing collaboration and reuse of code.

Distributed Version Control Systems

- When using a distributed version control system we don't necessarily have a central server for storing the data.
- Each and every developer may hold a clone for the main repository. Each clone is a full copy of the main repository.
- Each and every clone contains the full history of the collection and has the same functionality as the original repository.

Distributed Version Control Systems

- Each and every repository can exchange versions of the files it includes with other repositories by transporting these changes.
- We will usually have an up and running server that is always online.

Distributed Version Control Systems

- Using a distributed version control system we get the entire history of the files stored on each and every local machine and at the same time we get all local machines synced with the server.

What is Git?

- Git is one of the most popular distributed version control system.
- Git was developed by the Linux kernel development team and is used by many of today popular open source projects.
- Git was originally written in C. We can find Git implementations in other programming languages as Java, Ruby and Python.

What is Git?

- When using Git, each and every version is a snapshot of the files for a specific point in time.
- The collections of our project files and their complete history are stored in a repository.

Git Security

- Every file that goes to the Git repository is check-summed using an SHA-1 hash before been stored. Each time a file is retrieved Git validates it using the very same checksum that was calculated when been stored.

Git Performance

- Git was proved to be a very successful tool handling the files Linux includes.
- Git takes a snapshot of the entire set of files instead of storing the differences between each version. This simplicity ensures excellent performance even when dealing with huge number of files.
- Comparing with other versions control systems Git takes significantly less space.

Git Atomicity

❖ Git ensures that no data gets lost and no version mismatch happens dues to operations that were performed in a partial way.

Branching

- Git allows us to work a new different version of the collection of files.
- The new created branch is separated from its original collection. We can work on a new branch without effecting the collection from which it was created.

Merging

- Git allows us to merge two branches into one. This allows us to have one developer working on a new a feature while another is working on fixing a bug. Once the two complete their work we can merge the two branches into one.

The Working Tree

- The user works on a collection of files that originates from a certain point in time of the repository.
- This collection of files is known as the working tree. The user can create new files, change or delete.

The Staging Area

- Once we modify the working tree by creating a new file or changing an existing one we should go through two steps in order to persist these changes into the Git repository.
- We first need to add the selected files to the staging area. The second step would be to commit the changes we collected on the staging area to the Git repository.