Mouaz Ali
CSCI 3327

Git: A Distributed Version Control System

In the software development world, version control is a vital tool that helps with code changes and project stability, while allowing teamwork. Due to its effectiveness, adaptability, and powerful features, Git, a distributed version control system, has become the most widely used version control system today. Git comes with many different components, such as workflows, commits, pushes, pulls, merges, merge conflicts, and repositories. Because of such features, Git is now considered the foundation of modern software development.

When using Git to manage code, developers follow certain procedures and standards, which are referred to as workflows. Although there are different types of workflows, the Centralized Workflow is a common workflow for group work, as it uses a central repository to serve as the single point-of-entry for all changes to the project. In this workflow, the default branch is known as "main," where all changes are committed. In fact, no other branches besides main are necessary in this workflow. Needless to say, though, developers are free to select the workflow that best satisfies their demands, as each one serves different purposes.

In Git, commits are basically the building blocks of the development process, as they allow developers to trace the project's history. The standard Git commit process consists of: edit, stage, and commit. Commits ensure the fact that there's always an option to restore to a previous version of one's work in case something goes wrong. To put it into simpler words, we can think of commits like a snapshot of the project, allowing us to not only understand the work's past but also to come back to it at any particular moment, if needed.

Pushes and pulls are actions that have to do with remote repositories in Git. The process of sending local content or changes to a remote repository is actually called a push. In other words, pushing is a way of transferring your commits to a remote repository. Similarly, pulling is a way of fetching and downloading content from a remote repository, immediately updating the local repository to match that information. Both of these commands allow code changes to be synchronized between local and remote directories.

Merging is a way of putting a forked history back together again, allowing multiple lines of development to be combined into one. This is frequently used to incorporate modifications, bug patches, and new features into the main code. Merges are helpful because they allow multiple users to work on different components of a project. In the end, the contributions can be merged together. That way, developers don't have to wait on someone else to finish their part before they can contribute; it can be done simultaneously. When Git is unable to automatically resolve changes from two different branches, merge conflicts occur. When two users make changes to the same part of a file, Git may be unable to decide which should be allowed first. Git can only automatically merge changes if the commits are made on distinct lines or branches. In order to resolve a merge conflict, developers must intervene, manually selecting which changes to keep. The good thing is that it uses the same git status and git add commands for both generating commits and resolving merge conflicts.

Lastly, a key component of Git is a repository, which is a structure used to store all data. Repositories, being the central point for collaborative development, allow multiple developers to work on a project at the same time. Whether it's code, or slideshows, or documentation, everything can be kept in a Git repository, which can be hosted locally or remotely. As software development continues to evolve and grow, Git remains one of the best tools for any developer.