Git Overview

Git is an open source version control system. It is a widely-used tool for software development and information technology operations. When using Git, development teams must become familiar with some basic commands. Utilizing basic Git commands (such as init, clone, add, commit, push, pull, branch, merge, status, and many more), helps these teams maintain productivity and structure.

Git init is a command used to intitialize a Git repository. This is likely the first command one will run to start a new project. One might create a new directory for the repository or, alternatively, convert an existing directory into a Git repository. Git init is also called when running Git clone.

Git clone creates a copy of an already existing Git repository. This is useful to an individual for cloning one's own remote repository to a local repository and vice versa. Just as important is the ability to clone collaborative repositories, since Git is used by teams for task management.

Git add adds a change in a file from the working directory to the staging area. This is typically an in-between stage, as the goal is to eventually have all relevant changes saved in a repository. Files can be staged one at a time, or many at once. Git status is used to obtain the current state of the working directory and the staging area. Git add prepares the changes to be finally saved, or committed, to the repository.

Git commit is the command that finally saves changes to the repository. Git commit and Git add are often seen as the two most important Git workflow commands. Atlassian states that "Git commits are the core building block units of a Git project timeline"; therefore, it is essential that all Git users are familiar with this command and the purpose behind it.

Git push is the next step after Git commit, which finally pushes the changes to a remote repository. This allows for seamless collaboration between team members, as they would all theoretically have access to the remote repository. Git push and Git pull are important commands for remote-local/local-remote repository-to-repository communication.

Git pull downloads files from a Git repository and immediately saves them to the local repository. This allows team members to work on code locally, when someone else has already pushed their changes to the remote repository. Git pull utilizes Git fetch and Git merge. Before getting into merging, it is important to introduce Git branch.

Git branch has many functionalities for dealing with branches, such as creating, listing, renaming, and deleting branches. Atlassian describes a branch as "an independent line of development" in which users can test and develop code that they aren't ready to save to the main branch. After development in an individual branch is finished, users will use Git merge to integrate the changes into a master branch. Git merge doesn't always have to be used this way, but it is the most common use case.

Git checkout is commonly used for "undoing" changes. It does not erase commit history, but it can be used to revert a repository to a previous state.

Although these commands are the foundation of Git version control, users might not call them directly. They may use the GUI that comes with popular Git repository hosting services such as Github, GitLab, Bitbucket, or a hosting service created for a specific organization. Nevertheless, it is important for all Git users to understand what each command means.