Logan Esch
Technical Description
19 September 2014

What is Git

Git is an open source version control system. Think of it as manually running a backup on critical files every time an important change is made. This is desirable for a number of reasons including the frighteningly common occurrence of needing to modify half a dozen files in order to test a fix. One could cowboy up, make the changes on the fly, cross their fingers and hope that nothing is now broken. A wiser developer would make a copy of these files before modifying them to be able to quickly revert changes when things go awry. What if there is an uncovered edge case and the code does not need to be reverted until a month later? It's already a huge workload to copy files by hand, but having to keep track of each file in case problems arise from it months down the road seems like a gross misallocation of development time. Git was designed to make these situations manageable. Git allows developers to commit snapshots of files as they are changed, string them along in a history, and revert any combination of the files to any point in that history, at any time, all in a few keystrokes.

How is it Useful

What if a team has one developer working on a new feature and one working on a bug fix? Git supports having multiple versions of the source files that can developed and tested independently of each other and then be merged together later. Often this merge is completely automated. What the hard drive with all of the new features he had been working on for the past month meets an untimely end? With remote repositories in the cloud Git can prevent setbacks from all kinds of mishaps. The true power of Git though comes from its facilitation of collaboration on projects of any scale. Git allows code changes to be easily shared with others for review, testing, instruction or production. It allows large groups of people to be working on the code base at once and for their changes to be organized and coordinated so that their work results in better code. All of these benefits make Git one of the most crucial skills for a software developer to be familiar with.

Commits

The base unit in Git is a commit. A commit is a snapshot of how one or more files differ from the last commit in those file's history. When a file is committed, Git performs several actions. First it performs a diff of the file against the previous version of it. A diff is essentially subtracting the old version from the new, yielding the changes in between the versions. A hash of the diff is then generated using the SHA1 hashing algorithm. This hash is used to distinguish commits from one another. Commits contain a pointer to their ancestor commits. This means each commit knows which commit immediately proceeded it. The final part of a commit is the commit message. This is a message added by the author of the commit that should describe what changes the commit contains.

Logan Esch
Technical Description
19 September 2014

Branches

A history of commits in Git is called a branch and the default branch is 'master'. Branches can be checked out, created, deleted or merged. Checking out a branch plays all of the diffs of all the commits in that branch on the files that the user previously had checked out. This causes the users files to mirror the files whenever the last commit on the branch was made. When a new branch is made, a new commit is created that contains all of the currently checked out files. Deleting a branch also deletes the commits it contains, unless those commits have already been merged into another branch. Merging is one of the most useful features of Git. Merging one branch into another creates a new commit belonging to the branch that is checked out. This commit contains the previous commit on the branch and all of the changes from the branch being merged. If a file has been changed in both branches, Git is often still able to merge them seamlessly. Merge conflicts only occur if the same lines have been modified. If this happens, Git will pause before the automatic commit and alert the user after inserting both changes side by side into the file to let the user deconflict them. After manually merging the conflicted areas the user then must finish the commit.

Branches allow for different features or versions to be worked on independently of each other.

Repositories

Repositories or repos are the outer container of Git. They contain all the branches. The first thing Git does when a user wants to use it is initialize the repository, or create a new one if one does not exist. One of the most powerful features of Git is the use of remote repositories, which are ones that exist on a system other than the users. Remote repositories allow users to store copies of their projects on the cloud either privately or publicly. Private repos allow a user or group of users to collaborate on a project privately. Public repos are open to everyone to view, use and contribute. They allow people from around the world to collaborate on projects or use other's projects in their own work. Managing a project with even two collaborators can be a challenge. Git provides several ways to help manage a repository. The most basic is cloning, which is copies the master branch of a repository onto the users local system. After cloning a repository to their system a user can make changes and then push their commits up to a remote branch so others can see them or to simply back them up. If a remote branch contains different commits than the local one, the push will be rejected until the user has merged in the commits that they are missing. This can be done through a pull. A pull first compares the remotes most recent hash against its own. If the hashes differ, Git will fetch all the missing commits from the remote repository and merge them into the current branch. Often users who wish to contribute to repositories won't have direct push access to them. This is to prevent malicious or naïve users from breaking things. Instead users may fork a repository and have their own copy of it to make changes with. When a user has a change they would like to contribute to another repository they can create a pull request from their branch against the other repository. Git will then show a diff of the users proposed changes and other contributors can vet them.