Git Pocket Guide

What Is Git?

Git is a tool for tracking changes made to a set of files over time, a task traditionally known as “version control.” Although it is most often used by programmers to coordinate changes to software

source code, and it is especially good at that, you can use Git to track any kind of content at all. Any body of related files evolving over time, which we’ll call a “project,” is a candidate for

using Git. With Git, you can:

* Examine the state of your project at earlier points in time
* Show the differences among various states of the project
* Split the project development into multiple independent

lines, called “branches,” which can evolve separately

* Periodically recombine branches in a process called “merging,”

reconciling the changes made in two or more branches

* Allow many people to work on a project simultaneously,

sharing and combining their work as needed

Git is the technology behind the enormously popular “social coding” website GitHub, which includes many wellknown open source projects.

This is called a “root commit,” and most often, there is only one root commit in a repository—the

initial one created when the repository was started. However, you can introduce multiple root commits if you want; the command git checkout --orphan does this. This incorporates multiple independent

histories into a repository, perhaps in order to collect the contents of previously separate projects (see “Importing Disconnected History” on page 154).

Author versus Committer

Cherry-picking carries forward the author information from the original commit, while adding new committer information. This preserves the identification and origin date of the changes, while indicating that they were applied at another point in the repository at a later date, possibly by a different person.

**$ git log --format=fuller**

Other operations that do this are git rebase and git filterbranch; like git cherry-pick, they too create new commits based on existing ones.