Introduction to Revision Control and Source Code Management Plus A Super Tiny Git Reference Guide

Audience: This is a basic high-level introduction to the topics of revision control and source code management systems in general and to the git tool specifically for those largely unfamiliar with such things.

What are Revision Control and Source Code Management?

Code generally only grows in size and complexity over time—more methods or functions, more imported libraries, more calls, more files. Even if you're a perfect programmer, at minimum you probably would like to version your code in order to roll back to known points of functionality. Maybe your product manager figured out that customers only want features A–G but not H and I, and so you'd love to easily get back to that previous version of your work.

In addition to simple versioning, you and those with whom you work may need to support multiple variations of a code base, merge code changes across those variations, correlate bugs to file histories, make use of multiple versions of a library, regularly release production builds, and more. In order to do such things, you'll probably want to easily navigate a history of your code tree, intelligently apply patches, resolve change conflicts among developers, link to other code bases, and create labelled release snapshots. Surely, you'd also like to grant and revoke developer access, create automated backups of code, and much more.

All of this is where Revision Control and Source Code Management (RC and SCM) come in. They are the ringmasters in the software development circus that let you effectively manage multiple developers working on multiple text files over multiple spans of time.

What is Git?

Git—surprise, surprise—is a revision control and source code management system. It was born within the Linux community for sake of managing Linux kernel development following troubles with a commercial tool. Reportedly, its name is not a play on "get" but is taken from British slang. Git is among the third and latest generation of RC and SCM tools and is one of the most popular in that tier. Other tools like git include Bazaar and Mercurial. Git should not be confused with Github, a popular, commercial, repository service wrapped around the use of git.

Unlike previous generations of tools built around a distinct server and client, git is both simultaneously and is therefore a distributed revision control system (DRCS). You can think of a DRCS as peer-to-peer revision control. In practice, projects tend to orient themselves around a single “server” (e.g. Github) but this is not needed. Git works on top of many existing network protocols including HTTP and FTP, among others, and is backwards compatible to a limited degree with some previous generations of RC and SCM tools. A git repository is browsable through a web interface out of the box.

Git thinks about your source code as a complete and self-contained file system. Third generation tools like git rely on the concept of changesets for managing code changes among multiple developers. A changeset is an atomic collection of changes representing the transformation from one known state of the code file system to another. Previous generations of tools worked at the individual file level only and readily exposed the limitations of this approach. In using git, each developer has a local and complete copy of the code’s entire development history. Changes made by the local developer and remote contributing developers are both treated as simple branches from baseline code and are then available to be merged in to create a new up-to-date baseline.

The benefits of a system like git are many. Some of the principal benefits are: fast server-less operations against a local repository, freedom from network interruptions, multiple complete copies shared among project contributors (i.e. backups), and ease of working with large projects.

Availability

Git is freely available and officially supported as a command line tool for four platforms: Linux, OS X, Windows, and Solaris. In addition to the self-contained downloads available from the git project itself, you can also use your favorite package manager on Linux and OS X to install git.

Free and for-pay GUI git tools are available for Linux, OS X, and Windows.

Tiny Quickstart Reference Guide for This Class

The power of git and its brethren means that they can be a bit complex to work with at times. What is gained in solving many problems of earlier systems has a certain price in conceptual complexity and command structure. Some will debate this point, but for many developers it’s a big leap from popular earlier tools to git and the like.

Thankfully, the most common workflows of git are pretty easy to grasp. And for the purposes of your class and project you will almost certainly never need anything but the common simple git workflows.

Git Commands Needed for This Course

Clone

The clone command copies from a specified repository source a complete version of that repository to your local environment. Upon doing this, your local repository is a peer to the referenced source in the Distributed Revision Control System model git employs. Because of permission settings of the repositories you will use in this course, you will not be able to push your changes to the source repository (because this would break the examples and/or make your work available to every other student after you).

```
git clone url
```

Pull

From within your local git repository directory, the pull command fetches the latest changes from the repository you originally cloned and merges them with your local repository.

```
git pull
```

All Git Commands

To give you a basic feel for git in a very general sense, all of its commands are grouped and listed below. For in-depth documentation on these commands, see the References & Resources section at the end of this document.

Getting and Creating Projects

init, clone

Basic Snapshotting

add, status, diff, commit, reset, rm, mv, stash

Branching and Merging

branch, checkout, merge, log, tag

Sharing and Updating Projects

fetch, pull, push, remote

Inspection and Comparison

log, diff

References & Resources

- An overwhelming comparison matrix of revision control tools
- Version Control by Example
  ↳ Extensive introduction based in concepts and practices rather than specific tools
- Got 15 Minutes? Learn Git.
  ↳ Guided introduction to git with an interactive web-based terminal
- Just a simple guide for getting started with git. No deep shit ;-)
- Git Concepts Simplified
- How to Think Like Git
- In-depth Git Tutorial
- Comprehensive Git Reference