What is Source Control?

Learn about “source control” and its uses.

We'll cover the following

Source control

If you’re not familiar with source control, it solves a simple problem: how do you and your colleagues keep track of changes in your codebase?

You might start by:

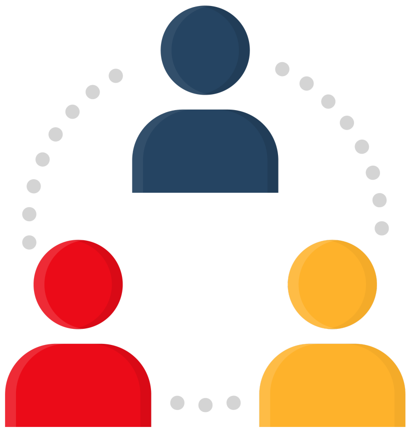
- Sending each other updated files
- Emailing each other when files change
- Keeping tar files in a central location with version numbers

Tracking file changes

However, whenever the project scales in size, you encounter problems:

- If more developers are involved, then there is a communication overhead to track all the changes.
- If there are multiple projects running concurrently on the same codebase, then tracking all the changes will get complicated.

- If multiple developers are working on the same files at the same time, then we need something to coordinate the changes.



Coordination between developers

A source control tool is a system that helps manage that complexity.

It’s a database of files and the histories of their states. Like a database, you have to learn the necessary skills to work on it and to get the full benefit.

I’m old enough to remember a time when people complained about using source control! These days, NOT using source control for projects is almost unheard of (but it still happens).

Life before Git

Before Git existed, there were (what might be called) traditional source control tools.

Traditional source control

Traditional source control tools, such as CVS and SVN, had a centralized architecture. You communicated with a server that maintained the state of the source code. This could mean several things:

- The source control database could get very big
- The history could get very messy

- Managing your checkouts of code could get complicated and painful

In the old world, if you *checked out source code*, then that was a copy of some code that was inferior in status to the centralized version.

As far as the user was concerned, the code was in one of two states:

- Local changes ("dirty")
- Committed

Committed was synonymous with "pushed to the server." My local changes could not be shared with anyone else until I committed and pushed them to the server.

Git

Git, by contrast, is fundamentally distributed. Each Git repository is a full copy of each Git repository it is copied from. It is not a "link" to a server or a "shadow" copy of another repository. You can make reference to the origin repository, but you do not have to do that. All code databases (in Git, CVS, or SVN) are known as **repositories**.

Git logo

Now remember this because It will be repeated often:

ALL GIT REPOSITORIES ARE BORN EQUAL!

Git was created so that people could work on the Linux kernel across the globe and offline. So there is no concept of a central server that holds the "golden" source. Instead, people maintain their own source code database (i.e., their own repository) and reconcile, copy from, and integrate with other repositories.

Linus Torvalds (the creator of Git and Linux) likes to joke that he has made the Internet his backup system.

Back
How the Course Works

Next
The Four Phases of Git Content

✓ Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/what-is-source-control__introduction-to-git__learn-git-the-hard-way)