GitHub

# The history of git

There are five important version control systems that predate Git that I want us to look at. There've been others but these are some of the most popular and the most influential and I think that they can help us to better understand Git. The first of these is called SCCS for Source Code Control System. Now it wasn't the first, but it was the first to become popular. It was released in 1972 and was developed by AT&T and it was bundled free with the Unix operating system. Now Unix was also free, and therefore, Unix spread quickly to places like universities and SCCS went along with it. Universities taught their students how to do version control using SCCS so when they left the university to go work in jobs, the version control system they were familiar with and that they took with them was SCCS. So, you can see why it became very popular. We talked about primitive version control earlier. That you might have a file like a budget, and you would save version one of that file, version two, and version three, giving them different file names each time. Now when you do that, you're saving the full document three different times. That's not a very efficient way to do it. What SCCS does is it keeps the original document, but then instead of saving the whole document a second time, it just saves a snapshot of what the changes were. So if you want version five of a document, you just take version one and apply four sets of changes to it to get to version five. That's a much more efficient way to store the changes over time. So SCCS stayed dominant until the early '80s when RCS was developed, Revision Control System, and it just made lots of improvements over SCCS. For one thing, it was cross-platform, whereas SCCS was Unix only. With the rise of the personal computer, it was important to have a version control system that would also work on PCs. It was also more intuitive, had a cleaner syntax with fewer commands, and more features. Most importantly, it was faster, and a lot of the speed increase came from the fact that it used a smarter storage strategy than SCCS. Remember SCCS stored the original file and then kept track of all the changes to that file that went after it. RCCS flipped that around so it kept the most recent file in its whole form, and if you wanted to go backwards in time and you wanted previous versions, then you applied the change snapshots to go in reverse. If you think about it, that's a lot faster because most of the time, what we want to work with is the current document. With SCCS, if we wanted the current document and there were 20 sets of changes, you had to pull up the original and then wait while 20 sets of changes were applied. With RCS, we can just bring up the current file, and it's already stored in its full state. One of the problems with both SCCS and RCS was that they only allowed you to work with an individual file one at a time. So you could track changes in a single file but not in sets of files or in a whole project. CVS, or Concurrent Version System, allowed you to do that. Now the real innovation in CVS is not just the fact that you can work with multiple files, it's the concurrent part. The fact that we can have a place where we can store our code called the code repository and you can put that on a remote server and more than one user can work on the same file at the same time. They can work concurrently. With previous versions, only one person could work with a file at a single time. So CVS adds a lot of features for users to be able to share their work and be able to update their file with changes that other people have made and placed in the remote repository. The idea of working with remote repositories was further improved upon with a Apache Subversion, or SVN for short. SVN was faster than CVS and it allowed saving of non-text files, like images, where CVS couldn't do that. Most importantly, the big innovation of SVN was that it was tracking not just changes to single files or to groups of files but watching what happened in a directory as a whole. Watching files and directories collectively and taking a snapshot of the directory, not just the files. Now that may seem like a small difference, but it's important. In CVS, you could talk about having version seven of some file. In SVN, you talked about having that file, as it appears in revision seven. SVN was able to track the history of directories. For example, CVS had a hard time if you renamed a file. SVN though, would track that change with no problem. If you add a file, remove a file, rename it, it's watching the directory as a whole to see what happens and taking a snapshot of that where CVS was just looking at a collection of individually named files. CVS would also update files one at a time as it went to either apply or read back changes. SVN would instead do that transactional commit and apply all of the changes that happened to the directory or to none of them at all. The snapshot was bigger than just the individual files. It was an entire directory or an entire set of changes that were happening to that directory at one time. It's a subtle but important difference. Now SVN stayed the most popular version control system for a very long time. In fact, until Git came out. But there's one other version control system that I want us to look at that comes in between and that's BitKeeper SCM. It was a closed source proprietary source code management tool. That means that a company owned it and sold it the same way that Adobe sells Photoshop or Microsoft sells Word. One of the important features that BitKeeper had, and it wasn't the first to have it, but that is distributed version control. Before we get to that, let's talk a little bit more about this idea of being closed source, where all the other ones that we've been looking at for a while have been open-source. The community version of BitKeeper was free and had a few less features and some usage restrictions. There was the paid version of BitKeeper but there was also a community version that they gave away for free and that version was used for source code management of the Linux kernel from 2002 to 2005. It was controversial to use a proprietary SCM for the Linux kernel because the Linux kernel is an open-source project, no one owns it. Where the SCM is owned and controlled by a company. So many people objected saying well what if they changed the rules in the future? We're going to be stuck using this company's software. Well guess what? In April 2005, the community version stopped being free and all those predictions came true. So BitKeeper was never as popular as CVS or SVN but it's important with the creation of Git because of its connection to Linux. Because, in April 2005, when the community version stopped being free, that's the same point at which Git was born. Git was created by Linus Torvalds and you may recognize that name as the person who created Linux and still drives the development of the Linux kernel. When BitKeeper stopped being free, they needed an alternative for managing their source code. Linus looked around and he didn't like the other VCSs that were out there like CVS and SVN. He did like some of the concepts of BitKeeper but he thought he could do even better so he wrote a new version control system from scratch, and that was Git. Git is distributed version control like BitKeeper. We'll talk more about distributed version control in the next movie. It's also open-source and free which is great for us because it means that people like you and me can download it for free, use it for free, and there's no license fees or anything like that. It also means because it's open-source, the community can see the source code and contribute to it. They can submit bug fixes, add new features, all those benefits we get because it's an open-source project. It's also compatible with most platforms like Linux, macOS, and Windows, and it's faster than most other source code management tools, 100 times faster in some cases for some operations. It also has better safeguards built into it to guard against data corruption. We'll talk a bit more about that later. Now, these improvements all worked. Git became a big hit. As people discovered the power of distributed version control as they get used to all of Git's nice features, Git experienced an explosion in popularity. Now there's no official statistics on this, but to give you an example, GitHub launched in 2008 as a platform to host Git source code repositories. In 2009, there were over 50,000 repositories with 100,000 users. In 2011, just two years later, there were two million repositories with over a million users. By 2018, GitHub was very popular. It was purchased by Microsoft. And in 2019, there were over 57 million repositories and over 28 million users. So Git has definitely taken off. In the next movie, let's talk about distributed version control and see why that's such an important feature.