Chapter 3

Meet the Current Players in the Game: SVN, Git, and Mercurial

Here’s a brief rundown of how this chapter is going to work. First of all, I’ll explain why Subversion (SVN), Git, and Mercurial are so awesome and all the things they do best. Then I’ll introduce each one individually, and describe what they do and how they do it (sounds a bit like a version control dating game!). Finally, I’ll show you how to get each one set up on Windows, Mac, and Ubuntu. I’ll be keeping this format—following the SVN, Git, and Mercurial layout—in later chapters. This holds true for improving workflow right up to creating a server, and everything in between. In each case, I’ll cover SVN, Git, and Mercurial; I don’t have any favorites here.

The differences between then and now

As you may remember from Chapter 1, I’ve already gone through a brief history of version control, starting way back with SCCS in 1972. This one started everything by being the first version control system (VCS); things really have come a long way.

There are a few products you need to be aware of to help make sense of the main players as I introduce them, the first of which is Concurrent Versions System (CVS). As already explained, CVS is a centralized system which was released in 1986. It did a good job for the time, but it was by no means perfect. The good thing about its flaws was that they led to the development of Apache Subversion (SVN), which is one of the systems you’re about to learn a lot about.
The two other systems you’re going to be learning about, Mercurial and Git, both stemmed from the same time. Back in 2005, BitKeeper withdrew its free version from the market, meaning there was no longer an open source distributed control system out there—thus both Mercurial and Git were born!

An introduction to the lead players: SVN, Git, and Mercurial

Now it’s time to really get into the swing of things, and get to know these systems a little better. At the moment they are all just names, without any relation or context. To remedy that, I’ll dive straight in at the deep end and explain the differences between the applications and why that matters.

What’s the difference between them and why should I care?

The main difference between these three systems is that two of them are distributed systems and the other, SVN, is a centralized system. Which means you have the repository either on your machine (distributed) or in one central (and in most cases, remote) location (centralized). That’s the essence of it on a very basic level; I’ll be going into more detail about each one very soon.

Using a centralized system allows for better adoption because the learning curve is significantly less than its non-centralized competitors. The code is checked out of the repository, you make a few changes, maybe even add some files, then you check for updates and, provided there aren’t any conflicts, push the code back in. Simple. Although distributed systems can take a bit of getting used to, the benefits of taking the time to do so can be huge. They allow you to have more control over your revisions, a great speed increase, and great scalability. Each of the systems discussed here has its own pros and cons, which I’ll be addressing soon enough; you can decide which one suits your needs best.

Apache Subversion (SVN)

Apache Subversion first appeared on the scene in October 2000 after it was funded and developed by CollabNet, Inc. It’s a centralized version control system, so if you ever commit/update files, they always originate from the same central location. It means you can always collaborate with other developers, even those in remote locations. Since SVN usually runs over HTTP, it’s easy for most people to use since HTTP is a standard protocol and is allowed on most firewall setups. SVN comes with built-in merging tools for those times when people have been working in the same place in a file. It tries its best to merge the two versions of the file together, but if it can’t resolve the conflict itself, it gives the option to you. When a conflict does come up, you can sort it out using a multitude of methods and programs, which I’ll go over in a lot more detail later on in the book.

Apache Subversion is a bit of a mouthful if you’re talking about it on a daily basis, so it was given the shortened name of SVN after the command used to control it. (I think it’s a lot more catchy, don’t you?) During development, its developers would use SVN to keep a history of web pages, documents, source code, and a lot more. This, of course, means the developers had enough confidence in it to use it while it was still being developed.