High-Speed Tutorial

“Please make sure your seat backs are in their full, upright position and that your tray tables are stored. Flight attendants, prepare for take-off….”

What follows is a quick tutorial that walks you through some basic Subversion configuration and operation. When you finish it, you should have a general understanding of Subversion's typical usage.

The examples used in this appendix assume that you have *svn*, the Subversion command-line client, and *svnadmin*, the administrative tool, ready to go on a Unix-like operating system. (This tutorial also works at the Windows command-line prompt, assuming you make some obvious tweaks.) We also assume you are using Subversion 1.2 or later (run `svn --version` to check).

Subversion stores all versioned data in a central repository. To begin, create a new repository:

```
$ cd /var/svn
$ svnadmin create repos
$ ls repos
conf/  dav/  db/  format  hooks/  locks/  README.txt
$
```

This command creates a Subversion repository in the directory */var/svn/repos*, creating the *repos* directory itself if it doesn't already exist. This directory contains (among other things) a collection of database files. You won't see your versioned files if you peek inside. For more information about repository creation and maintenance, see Chapter 5, *Repository Administration*.

Subversion has no concept of a “project.” The repository is just a virtual versioned filesystem, a large tree that can hold anything you wish. Some administrators prefer to store only one project in a repository, and others prefer to store multiple projects in a repository by placing them into separate directories. We discuss the merits of each approach in the section called “Planning Your Repository Organization”. Either way, the repository manages only files and directories, so it's up to humans to interpret particular directories as “projects.” So while you might see references to projects throughout this book, keep in mind that we're only ever talking about some directory (or collection of directories) in the repository.

In this example, we assume you already have some sort of project (a collection of files and directories) that you wish to import into your newly created Subversion repository. Begin by organizing your data into a single directory called *myproject* (or whatever you wish). For reasons explained in Chapter 4, *Branching and Merging*, your project's tree structure should contain three top-level directories named *branches*, *tags*, and *trunk*. The *trunk* directory should contain all of your data, and the *branches* and *tags* directories should be empty:

```
/tmp/
  myproject/
    branches/
    tags/
  trunk/
    foo.c
    bar.c
    Makefile
    …
```

The *branches*, *tags*, and *trunk* subdirectories aren't actually required by Subversion. They're merely a popular convention that you'll most likely want to use later on.

Once you have your tree of data ready to go, import it into the repository with the *svn import* command (see the section called "Getting Data into Your Repository"):

```
$ svn import /tmp/myproject file:///var/svn/repos/myproject \
      -m "initial import"
Adding         /tmp/myproject/branches
Adding         /tmp/myproject/tags
Adding         /tmp/myproject/trunk
Adding         /tmp/myproject/trunk/foo.c
Adding         /tmp/myproject/trunk/bar.c
Adding         /tmp/myproject/trunk/Makefile
…
Committed revision 1.
$
```

Now the repository contains this tree of data. As mentioned earlier, you won't see your files by directly peeking into the repository; they're all stored within a database. But the repository's imaginary filesystem now contains a top-level directory named *myproject*, which in turn contains your data.

Note that the original */tmp/myproject* directory is unchanged; Subversion is unaware of it. (In fact, you can even delete that directory if you wish.) To start manipulating repository data, you need to create a new "working copy" of the data, a sort of private workspace. Ask Subversion to "check out" a working copy of the *myproject/trunk* directory in the repository:

```
$ svn checkout file:///var/svn/repos/myproject/trunk myproject
A    myproject/foo.c
A    myproject/bar.c
A    myproject/Makefile
…
Checked out revision 1.
$
```

Now you have a personal copy of part of the repository in a new directory named *myproject*. You can edit the files in your working copy and then commit those changes back into the repository.

- Enter your working copy and edit a file's contents.
- Run `svn diff` to see unified diff output of your changes.
- Run `svn commit` to commit the new version of your file to the repository.
- Run `svn update` to bring your working copy "up to date" with the repository.

For a full tour of all the things you can do with your working copy, read Chapter 2, *Basic Usage*.

At this point, you have the option of making your repository available to others over a network. See Chapter 6, *Server Configuration* to learn about the different sorts of server processes available and how to configure them.