Extracted from:

Pragmatic Version Control

using Subversion, 2nd Edition

Chapter 5

Accessing a Repository

In Chapter 3, *Getting Started with Subversion*, on page 28, we created a repository and learned how to access it via a file-based URL. This is great for a single user but doesn't really help a whole development team collaborate properly. In this chapter we'll discuss the three main ways you can make an existing repository available over the network, what they mean for a user accessing a repository, and the pros and cons of the various access mechanisms.

Appendix A on page 151 includes a guide for administrators who are installing, networking, and securing Subversion.

5.1 Network Protocols

After creating our sandbox repository, we used a *repository URL* to tell Subversion what we wanted to check out. This URL included both a definition of where the repository was and also what path inside the repository we were interested in. Once we had a working copy we didn't need to keep using the repository URL, since Subversion remembers where our working copy came from.

repository URL

Repository URLs are important whenever we want to directly access a repository (when we're creating branches and tags or merging big sets of changes, for example). Figure 5.1 on the following page shows how the URL for our sandbox repository is composed.

The first part of this URL is `file`. This specifies the *scheme* we're using to locate the repository, in this case the local

scheme

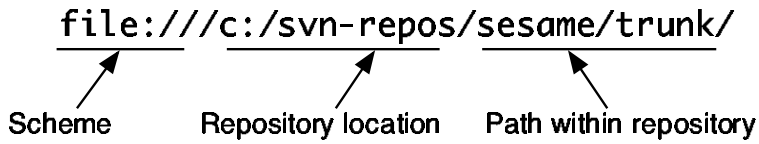


Figure 5.1: COMPONENTS OF A REPOSITORY URL

filesystem. The next part, `c:/svn-repos`, tells Subversion the repository database files are in a particular directory on the C: drive. Finally, `/sesame/trunk/` specifies the path within the repository that we're interested in.

Subversion supports a number of different schemes in repository URLs and even allows you to define custom extensions yourself. Each different scheme tells Subversion to access the repository via a particular network protocol. We'll start by looking at the simple `svn` protocol.

svn

The easiest way to network a repository is to use the `svn` scheme. Subversion comes with svnserve, a small server that listens for network connections, allows repository access over the network, and supports simple authentication of users. svnserve is probably most suitable for teams on a private LAN who want to get going quickly.

If an administrator (possibly you!) has used the instructions in Section A.2, *Networking with svnserve*, on page 153 to put the Sesame repository online, you can check it out by running

```
work> svn co svn://olio/sesame/trunk vizier
A  vizier/Number.txt
A  vizier/Day.txt
Checked out revision 7.
```

Success! We used the `svn` scheme to access a repository on a machine called `olio`, and we checked out the Sesame project to a new vizier working directory.

If you've tried playing with the working copy on your client machine, you might find that Subversion doesn't let you com-

mit any changes. For example, try adding a new data file, Month.txt, to the project:

```
vizier> svn add Month.txt
A         Month.txt
vizier> svn commit -m "Added month data"
svn: Commit failed (details follow):
svn: Connection is read-only
```

If this happens, your administrator has forgotten to enable write access to the repository (it's read-only by default). Get them to look at Section A.5, *svnserve*, on page 163 and set up some users. Once they've done this, you should be asked for a username and password when you try to commit a change:

```
vizier> svn commit -m "Added month data"
Authentication realm: <svn://olio:3690> sesame/trunk
Password for 'mike':
Adding         Month.txt
Transmitting file data .
Committed revision 8.
```

Subversion decided to try username `mike` because that's my username on the client machine. If this isn't right, just hit Enter at the password prompt, and Subversion will let you specify a different username.

svn+ssh

svnserve does a great job of getting a repository up on the network, but it has a couple of drawbacks. Firstly, although passwords are never transmitted in clear text over the network, the *contents* of your files travel unencrypted. Anyone who can sniff your network traffic can see what your files contain. This might be okay for a team all on the same LAN, but if you want to use the public Internet for accessing your repository it simply isn't secure. Secondly, passwords are stored in plain text in the server's conf directory and can only be changed by an administrator with access to the password file.

Subversion solves both of these security problems by leveraging the *Secure Shell* (SSH). If you're a Unix user, you might already have SSH infrastructure in place for connecting to your server. SSH employs strong encryption to protect the contents of a client-server session. It is widely used for administering servers over the Internet. Figure 5.2 on the next page shows how Subversion secures an `svn` connection using SSH.

Secure Shell

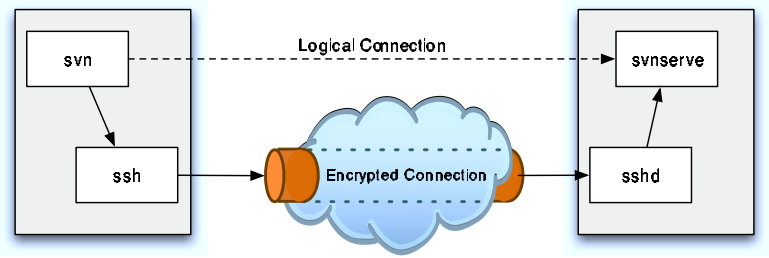


Figure 5.2: TUNNEL SUBVERSION OVER SSH

Subversion needs an SSH client installed on your machine in order for you to access a repository using `svn+ssh`. Unix users are likely to have SSH already installed, but if you're on Windows, you'll need to do a bit of work. Putty is an excellent SSH client and is available from `http://www.chiark.greenend.org.uk/~sgtatham/putty/`. Download plink.exe, and save it somewhere in your path; C:\Windows\system32 usually works. If you're using TortoiseSVN you don't need to worry about installing an SSH client since Tortoise comes with TortoisePlink.

Next you need to edit your Subversion client configuration settings. Windows applications store user-specific data inside a special folder, which changes location depending upon how your computer is set up and which version of Windows you're using. If you're not sure where your application data directory is, open a command prompt and run the following:

```
work> echo %APPDATA%
C:\Documents and Settings\mike\Application Data
```

Once you've found your application data directory, open the Subversion subdirectory, and edit the config file that's inside. Edit the section on tunnels so it looks like this:

```
[tunnels]
ssh=plink
```

You need to specify a `svn+ssh` scheme if you'd like Subversion to use SSH to protect your connections. If your server accepts SSH connections, try running

```
work> svn checkout  \
          svn+ssh://olio/home/mike/svn-repos/sesame/trunk  \
          princess
mike@olio's password:
A  princess/Month.txt
A  princess/Number.txt
A  princess/Day.txt
Checked out revision 8.
```

This looks just like the repository URL we used earlier with svnserve, except we changed the scheme to `svn+ssh`. If you're having problems accessing your repository, Section A.3, *Troubleshooting an SSH Connection*, on page 156 contains a guide to diagnosing the problem.

Subversion is now using SSH to open a connection to the server and authenticate you as a Unix user. Subversion uses the standard Unix user and group permissions to determine whether the user with which we connect has permission to access the repository. If you're using SSH public/private keys or an SSH agent to manage your credentials, the Subversion client automatically takes advantage of this, which might mean you don't get asked for a password at all.

Using `svn+ssh` is appealing if you already have SSH accounts for your users, because you can leverage all your existing infrastructure. The extra security lets you connect over the Internet without fear that someone might steal your Sesame project code and without all the hassle of setting up a full VPN. `svn+ssh` is a straightforward solution that should have you up and running pretty fast.

http

Subversion can also host a repository over the web by using the Apache web server. A special Subversion module, called `mod_dav_svn`, does the hard work and allows Subversion to share the web server with traditional web sites. Apache is highly configurable, and Subversion takes full advantage of its built-in security and scalability. You can host a repository using standard `http` and `https` and leverage any of the authentication mechanisms already supported by Apache.

You may have heard that Subversion *requires* Apache—this actually isn't true; neither `svn` nor `svn+ssh` need anything

extra to network your repository. Most prebuilt Unix packages have a dependency on Apache because they install all three networking options, which is where the misunderstanding comes from. Using Subversion with Apache is probably the most popular solution for sharing a repository over the Internet.

Apache provides a wealth of authentication options for users. From basic authentication using password files to integration with a Windows domain or an LDAP server, Apache is supremely flexible. You can even set up directory-based security, dividing your repository into read-only or even completely private sections. You can take advantage of standard SSL certificates for encrypting connections to the server and avoid firewall hassles by using standard web server port numbers.

To access a repository hosted by Apache on server `olio`, use the following command:

```
work> svn checkout  \
          http://olio.mynetwork.net/svn-repos/sesame/trunk   \
          sesame
Authentication realm: ... Subversion repository
Password for 'mike': ******
A  sesame/Month.txt
A  sesame/Number.txt
A  sesame/Day.txt
Checked out revision 8.
```

This particular repository requires an authenticated user even for read-only access. Subversion automatically tries username `mike`; if that's wrong, just hit Enter instead of typing a password, and Subversion will let you specify the username.

5.2 Choosing a Networking Option

All three network protocols for Subversion (`svn`, `svn+ssh` and `http`) offer different trade-offs in terms of ease of setup, security, and administration overhead. Which you choose will depend on what kind of infrastructure you already have, your security needs, and your familiarity with Apache.

It's important to note that the networking option you choose today doesn't have to be the one you stick with tomorrow. Networking a repository simply puts it on the network—you can change between svnserve and Apache (for example) as often as

you like. It's also possible to support multiple different access mechanisms *at the same time*, although you have to be careful with permissions.

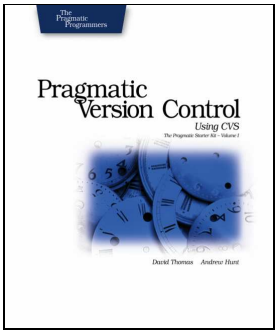
If your team is on a reasonably secure LAN, or even a larger network connected by a VPN, using the simple svnserve server and `svn` protocol is a quick way to get up and running with Subversion. You'll have some administrative overhead when adding new users or changing passwords, but this should be offset by the easy startup. Subversion 1.3 added directory-based authorization to svnserve making it almost as flexible as Apache for teams on the same LAN.

If you already have existing SSH infrastructure in place, using `svn+ssh` makes a lot of sense. You get strong crypto protecting your connections and can take advantage of all of the key-management and authentication options that SSH provides. Make sure your Unix administrator understands how groups, umasks, and sticky bits need to be set up before proceeding, though.

If you want to host a repository over the Internet, leverage Apache's wide range of authentication mechanisms, or simply play with the big boys and run a "real" server, using Apache to host your Subversion repository is the way to go. You'll be able to use SSL and client-server certificates for encryption and verifying you're really talking to whom you think you're talking to, and you'll be able to authorize users using a Windows domain, LDAP, or any other authentication mechanism that Apache supports. You'll also be able to be much more precise about which parts of a repository users have access to, by leveraging the `mod_authz_svn` Apache module. Using Apache on standard HTTP ports also means fewer holes need to be opened on your firewalls. Your network administrator will thank you for that.

Pragmatic Starter Kit

Version Control. **Unit Testing**. **Project Automation**. Three great titles, one objective. To get you up to speed with the essentials for successful project development. Keep your source under control, your bugs in check, and your process repeatable with these three concise, readable books from The Pragmatic Bookshelf.

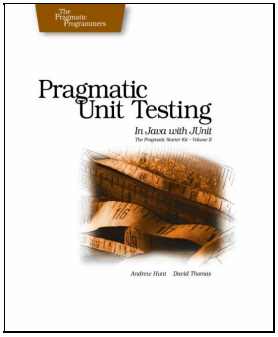


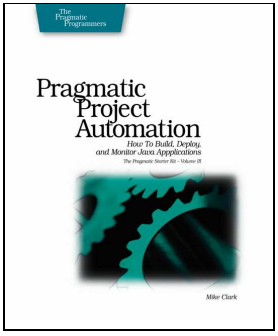
The CVS companion to this book • Keep your project assets safe—never lose a great idea • Know how to UNDO bad decisions—no matter when they were made • Learn how to share code safely, and work in parallel • See how to avoid costly code freezes • Manage 3^{rd} party code • Understand how to go back in time, and work on previous versions.

Pragmatic Version Control using CVS
Dave Thomas and Andy Hunt
(176 pages) ISBN: 0-9745140-0-4. $29.95

• Write better code, faster • Discover the hiding places where bugs breed • Learn how to think of all the things that could go wrong • Test pieces of code without using the whole project • Use JUnit to simplify your test code • Test effectively with the whole team.

Pragmatic Unit Testing
Andy Hunt and Dave Thomas
(176 pages) ISBN: 0-9745140-1-2. $29.95
(Also available for C#, ISBN: 0-9745140-2-0)





• Common, freely available tools which automate build, test, and release procedures • Effective ways to keep on top of problems • Automate to create better code, and save time and money • Create and deploy releases easily and automatically • Have programs to monitor themselves and report problems.

Pragmatic Project Automation
Mike Clark
(176 pages) ISBN: 0-9745140-3-9. $29.95

Visit our secure online store: `http://pragmaticprogrammer.com/catalog`

Pragmatic Starter Kit

Version Control. **Unit Testing**. **Project Automation**. Three great titles, one objective. To get you up to speed with the essentials for successful project development. Keep your source under control, your bugs in check, and your process repeatable with these three concise, readable books from The Pragmatic Bookshelf.

Visit Us Online

Pragmatic Version Control using Subversion
`pragmaticprogrammer.com/titles/svn`
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
`pragmaticprogrammer.com/updates`
Be notified when updates and new books become available.

Join the Community
`pragmaticprogrammer.com/community`
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
`pragmaticprogrammer.com/news`
Check out the latest pragmatic developments in the news.

Buy the Book

If you liked this PDF, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: `pragmaticprogrammer.com/\HomePageUrl`.

Contact Us

Phone Orders:	1-800-699-PROG (+1 919 847 3884)
Online Orders:	`www.pragmaticprogrammer.com/catalog`
Customer Service:	`orders@pragmaticprogrammer.com`
Non-English Versions:	`translations@pragmaticprogrammer.com`
Pragmatic Teaching:	`academic@pragmaticprogrammer.com`
Author Proposals:	`proposals@pragmaticprogrammer.com`