

# Subversion for SysAdmins

Contributed by David Torre

If you've ever needed to examine a previous version of a document-- be it application code, a UNIX configuration file, or a customer invoice, version control systems can save the day. Using Subversion, I'll show you how to build an enterprise-class version control system in under an hour.

{mospagebreak title=Overview}

## Benefits of Version Control

While most software configuration management systems boast a vast assortment of complex features, this article will focus primarily on the benefits of "versioning." As trivial as it may seem, keeping track of previous file and folder versions can be an invaluable feature to both developers and systems administrators alike. Having worked on the SysAdmin side of the fence for most of my career, I've encountered countless episodes whereby seemingly minor changes to system configurations led to adverse consequences. When changes to the system cause issues, you simply need to know "who edited what" as soon as possible. Thankfully, a versioning system does just that. By checking in a server's configuration directory (such as /etc for example), you now have a redundant copy of your server configuration, as well as several levels of revision history, any of which you may choose to revert back to in the event of chaos.

In a nutshell, the job of a version control system is to keep track of file and folder revisions throughout the lifespan of a project. Moves, adds, or changes to files and folders are recorded in the system, allowing one or more users to work concurrently on the same project. Internally, version control systems store only the differences (or changes) to files, rather than keeping full-blown redundant copies of each revision. This allows for efficient storage and prompt synchronization between clients and repositories.

Although there are many software configuration management and version control systems available to choose from, this article will focus on building a version control system based on the popular open source application-- Subversion

Depending on your requirements, there are several ways in which one may access a Subversion repository. Methods of repository access include:

- Local file system &ndash; Subversion client performs versioning using local directories

-  
Proprietary &ldquo;SVN&rdquo; Protocol &ndash; TCP-based Subversion calls over clear-text or SSH

-  
WebDAV / DeltaV &ndash; Subversion operations occur via DAV-enabled HTTP(S)

For this article, I'll demonstrate how to deploy SVN using a DAV-enabled Apache server. This approach provides an Internet-accessible client/server architecture, and allows communications to occur over standard HTTP which will work within most firewalled environments. In addition to the SVN repository, we'll also install an optional ViewVC web front-end for simple read-only access to your repositories.

## Assumptions

This article assumes you're working with a Red Hat-based system. Although the steps conveyed within this article may easily be modified for use with other Linux or UNIX-based systems, the article will utilize features such as yum and chkconfig, which are often found in Red Hat-like Linux distributions.

{mospagebreak title=Software Installation}

## Installing Apache with mod\_dav\_svn

We'll begin by using Yum to install Python, Apache, and mod\_dav\_svn. This is done by typing the following, as root:

```
yum -y install httpd mod_dav_svn subversion python
```

Depending on your Yum repository configuration, you may or may not grab the latest versions of Subversion and mod\_dav\_svn. If you plan on using the web front-end ViewVC, ensure Python is at version 2.0 or greater, and that Subversion is at 1.2.0 or greater. For example:

```
[root@server ~]# rpm -qa | grep subversion && python -V  
subversion-1.1.4-2.ent  
Python 2.3.4
```

As you can see, my Python installation is fine, but Subversion is slightly out of date. Again, if you plan on using ViewVC, Subversion must be at v1.2, which means our mod\_dav\_svn must also be updated. Therefore, let's grab the latest Subversion and mod\_dav\_svn packages from Dag Wieers:

```
wget http://dag.wieers.com/your-distro/subversion-1.2.1-0.1.2.el4.rf.i386.rpm
wget http://dag.wieers.com/your-distro/mod_dav_svn-1.2.1-0.1.2.el4.rf.i386.rpm
```

Now upgrade the two RPMs we just downloaded:

```
[root@server ~]# rpm -Uvh *.rpm
Preparing...      ##### [100%]
1:subversion      ##### [ 50%]
2:mod_dav_svn     ##### [100%]
```

Our Python and Subversion packages should show be at suitable version levels for use with ViewVC:

```
[root@server ~]# rpm -qa | grep subversion && python -V
subversion-1.2.1-0.1.2.el4.rf
Python 2.3.4
```

## Apache Startup

Now that the Apache server has been installed, we're going to go ahead and start it, and ensure it auto-starts upon subsequent reboots. However, you must first set the `ServerName` parameter in: `/etc/httpd/conf/httpd.conf` to reflect your machine's hostname or IP. You may also want to update the `ServerAdmin` parameter as well. My `httpd.conf` appears as follows:

```
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address anyway, and this will make
# redirections work in a sensible way.
#
ServerName svn.unixvoodoo.com:80
```

Now to start Apache, and ensure it auto-starts after each reboot:

```
[root@server ~]# service httpd start
Starting httpd:          [ OK ]
[root@server ~]# chkconfig httpd on
```

## ViewVC Installation (Optional)

You should now head over to <http://www.viewvc.org/download.html> and grab the latest copy of ViewVC. For our exercise, I'll download the latest version to a temporary directory for extraction, then run the viewvc-install script. Simply hit enter to accept all defaults:

```
[root@server viewvc-1.0.4]# ./viewvc-install
This is the ViewVC 1.0.4 installer.
It will allow you to choose the install path for ViewVC. You will now
be asked some installation questions. Defaults are given in square brackets.
Just hit [Enter] if a default is okay.
Installation path [/usr/local/viewvc-1.0.4]:
DESTDIR path (generally only used by package maintainers) []:
Installing ViewVC to /usr/local/viewvc-1.0.4:
...removed...
ViewVC file installation complete.
Consult the INSTALL document for detailed information on completing the
installation and configuration of ViewVC on your system. Here's a brief
overview of the remaining steps:
1) Edit the /usr/local/viewvc-1.0.4/viewvc.conf file.
2) Either configure an existing web server to run
   /usr/local/viewvc-1.0.4/bin/cgi/viewvc.cgi.
   Or, copy /usr/local/viewvc-1.0.4/bin/cgi/viewvc.cgi to an
   already-configured cgi-bin directory.
   Or, use the standalone server provided by this distribution at
   /usr/local/viewvc-1.0.4/bin/standalone.py.
```

At this point, all required software has been installed. In the next section, we'll walk through the initial configuration and security of your newly installed Subversion repository.

{mospagebreak title=Repository Configuration}

## ViewVC Configuration

Fortunately, the ViewVC configuration is fairly straight forward. Simply open up the viewvc.conf file (locate at /usr/local/viewvc-x.y.z/viewvc.conf by default), then modify the root\_parents, the default\_root, and finally the address parameters as follows:

```
root_parents = /var/www/svnroot : svn
# this is the name of the default root
# (ignored when root_as_url_component is turned on)
default_root =
address = svn-admins@your-organization.com
```

Don't worry about the svnroot directory yet, as we will be creating it shortly. The last thing we need to do here is tell ViewVC where to look when attempting to authenticate web users. Create a viewvc.conf Apache include file (perhaps at

/etc/httpd/conf.d/viewvc.conf) and append the following to the the file:

```
ScriptAlias /viewvc /usr/local/viewvc-1.0.4/bin/cgi/viewvc.cgi
<Location />
AuthName "SVN Repository Authentication"
AuthType Basic
AuthUserFile /var/www/svnroot/svn-accounts
Require valid-user
</Location>
```

ViewVC should now be installed, configured, and ready to authenticate web users.

## Apache Configuration

Since Apache will be serving up our SVN repositories via WebDAV, obviously WebDAV itself must be enabled. Although WebDAV is enabled by default on many Apache installations, we'll double-check by opening the main httpd.conf file (locate at /etc/httpd/conf/httpd.conf on Red Hat systems), and locating the &ldquo;LoadModule&rdquo; section. Once you've located this section, ensure you have the following line set:

```
LoadModule dav_module modules/mod_dav.so
```

If you installed mod\_dav\_svn via yum/rpm, you'll have an Apache include file located at: /etc/httpd/conf.d/subversion.conf. Go ahead and open this file, and ensure you have the following lines set:

```
LoadModule dav_svn_module modules/mod_dav_svn.so
LoadModule authz_svn_module modules/mod_authz_svn.so
```

We'll now go ahead and create the root-level directory structure for our SVN repository, starting at /var/www/svnroot. All Subversion repositories, as well as security database files, will live at this location. (Note that lines beginning with # are comments)

```
#Create SVN root
[root@server viewvc-1.0.4]# mkdir /var/www/svnroot
#Create username/password database
[root@server viewvc-1.0.4]# touch /var/www/svnroot/svn-accounts
#Create permissions access file
[root@server viewvc-1.0.4]# touch /var/www/svnroot/svn-acl
[root@server viewvc-1.0.4]# chown -R apache.apache /var/www/svnroot/*
```

We now have a working root Subversion location, waiting for new projects!

## Creating a New Repository

Creating a new repository is quite simple. Navigate to your SVN root directory (`/var/www/svnroot`) and invoke `svnadmin create` command, using your project name as a final argument. For example, if we wanted to create a repository named `&ldquo;oracle-ERP&rdquo;`, we'd issue the following command:

```
[root@server /]# cd /var/www/svnroot/  
[root@server svnroot]# svnadmin create oracle-ERP
```

We now need to make sure the newly created `&ldquo;oracle-ERP&rdquo;` repository is writable by the web server user. For Red Hat systems, this is typically user `'apache.'` To set the user and group ownership to Apache (or whoever your web server currently runs as), type the following as root:

```
[root@server svnroot]# chown -R apache.apache oracle-ERP/
```

Now that we've added our new repository, we need to tell `mod_dav_svn` that it exists. This is done by opening your `subversion.conf` Apache include file (located by default at: `/etc/httpd/conf.d/subversion.conf`) and appending the following to the end of the file:

```
<Location>  
  
DAV svn  
  
SVNPath /var/www/svnroot/oracle-ERP  
  
AuthType Basic  
  
AuthName "oracle-ERP"  
  
AuthUserFile /var/www/svnroot/svn-accounts  
  
Require valid-user  
  
AuthzSVNAccessFile /var/www/svnroot/svn-acl  
  
</Location>
```

Finally, let's restart Apache for our changes to take effect:

```
[root@server conf.d]# service httpd restart
```

## Removing Repositories

The removal of a given repository is achieved simply through removing the directory structure located at the SVN root location, as well as any configuration directives in your subversion.conf include file. Specifically:

- Remove repository entry from /etc/httpd/conf.d/subversion.conf
- Delete the repository directory (and all sub-directories) from /var/www/svnroot

## Adding Users and Groups

Since you'll most likely need your Subversion repositories to be password-protected, we'll walk through the creation of two users, and one group. Go ahead and open the file /var/www/svnroot/svn-acl, and add your organization's users and groups:

```
#
# specify groups here
#
[groups]
team-it = dave, jen
#
# Group "team-it" and a contractor named "mike" have read/write access to the
# squid-config repository. All others have read-only access
#
[squid-config:]
@team-it = rw
mike = rw
* = r
#
# Group "team-it" has read-write access to the oracle-ERP repository.
# All others have no access
#
[oracle-ERP:]
@team-it = rw
```

Now that we've established permissions on a couple SVN repositories, let's create passwords for the three users we referenced in svn-acl. This is done by invoking htpasswd to build our user database file, located at: /var/www/svnroot/svn-accounts:

```
[root@server svnroot]# htpasswd -m /var/www/svnroot/svn-accounts dave
New password: daves-password
Re-type new password: daves-password
```

```
Adding password for user dave
[root@server svnroot]# htpasswd -m /var/www/svnroot/svn-accounts jen
New password: jens-password
Re-type new password: jens-password
Adding password for user jen
[root@server svnroot]# htpasswd -m /var/www/svnroot/svn-accounts mike
New password: mikes-password
Re-type new password: mikes-password
Adding password for user mike
[root@server svnroot]#
```

## Removing Users and Groups

Removing users and/or groups is done via editing the `/var/www/svnroot/svn-accounts` and `/var/www/svnroot/svn-acl` files using your favorite text editor.  
At this point, you have a working Subversion repository which you may use either via your favorite Subversion client{link}, or via the ViewVC interface, located at <http://your-server.com/viewvc>.

{mospagebreak title=Using your Repositories}

## Using your Repositories

By now, you should have a bare-bones Subversion server, accessible via a WebDAV-enabled Apache server. Although we created the mock "oracle-ERP" repository in the previous section, we'll now direct our attention to a slightly more IT-specific example-- checking in an Linux web server's entire `/etc/httpd` directory into its own repository.

Start off by creating a new repository, named "web1-etc"; This may be done either by following the directions from the previous section, or using a the provided `mkrepo.sh` shell script:

```
#!/bin/sh
if ! [ $1 ]
then
echo "Usage: mkrepo.sh repo-name"
else
svnadmin create /var/www/svnroot/$1
chown -R apache.apache /var/www/svnroot/$1/*
<location>
DAV svn
SVNPath /var/www/svnroot/$1
```

```
AuthType Basic
AuthName "$1"
AuthUserFile /var/www/svnroot/svn-accounts
Require valid-user
AuthzSVNAccessFile /var/www/svnroot/svn-acl
" >> /etc/httpd/conf.d/subversion.conf
service httpd graceful
```

We now need to set permissions for the new repository in /var/www/svnroot/svn-acl:

```
#
# Group "team-it" has read-write access to the web1-etc repository.
# All others have no access
#
[web1-etc:]
@team-it = rw
```

Now that we have a repository set aside for the server &ldquo;web1,&rdquo; let's check-in our Apache directory to the newly created repository:

```
[root@server /]# cd /
[root@server /]# svn co http://svn.your-domain.com/web1-etc /
Authentication realm: SVN Repository Authentication
Password for 'root': [Just hit enter]
Authentication realm: SVN Repository Authentication
Username: dave
Password for 'dave': [enter your password]
Authentication realm: SVN Repository Authentication
Checked out revision 0.
```

Now that we have a working version checked-out, we can need to tell SVN which files and folders we want to keep track of. Although you may theoretically check-in all of /etc (or most of / for that matter), we'll restrict our repository to /etc/httpd and all subdirectories. Our first step is to add /etc, but without all the other unnecessary subdirectories. This is done by passing the -N flag to the &ldquo;add&rdquo; command:

```
[root@server /]# svn add -N /etc
A    /etc
```

We now may proceed to adding the Apache configuration directory:

```
[root@server /]# svn add /etc/httpd/
A    /etc/httpd
A    /etc/httpd/logs
A    /etc/httpd/run
A    /etc/httpd/conf.d
A    /etc/httpd/conf.d/viewvc.conf
A    /etc/httpd/conf.d/subversion.conf
A    /etc/httpd/conf.d/README
A    /etc/httpd/conf.d/welcome.conf
A    /etc/httpd/conf
A    /etc/httpd/conf/httpd.conf
A    /etc/httpd/conf/magic
A    /etc/httpd/modules
[root@server /]#
```

Thus far, we've created a repository to hold our Apache configuration, checked out the latest revision (basically revision 0, with nothing in it), and told SVN which files and folders we want to include in the repository. From this point forward, we can start &ldquo;checking in&rdquo; our configurations at regular intervals. This is done with a basic &ldquo;commit&rdquo; or &ldquo;ci&rdquo; command:

```
[root@server /]# cd /
[root@server /]# svn ci -m "Initial check-in by Dave..."
Adding    etc
```

```
Adding    etc/httpd
Adding    etc/httpd/conf
Adding    etc/httpd/conf/httpd.conf
Adding    etc/httpd/conf/magic
Adding    etc/httpd/conf.d
Adding    etc/httpd/conf.d/README
Adding    etc/httpd/conf.d/subversion.conf
Adding    etc/httpd/conf.d/viewvc.conf
Adding    etc/httpd/conf.d/welcome.conf
Adding    etc/httpd/logs
Adding    etc/httpd/modules
Adding    etc/httpd/run
Transmitting file data .....
Committed revision 1.
```

It's important to note that newly introduced files will not be added to your working copy of the repository by default. If you create a new Apache sub-config file (such as `/etc/httpd/conf.d/some-new-module-config.conf`) you'll need to explicitly add this file using `&ldquo;svn add [filename]&rdquo;`; Alternatively, you may cron a re-add of everything in `/etc/httpd` perhaps once per day using something similar to:

```
svn status /etc/httpd | grep ^\? | cut -c8- | xargs svn add
```

From here on out, you may wish to check-in your Apache directory at schedule intervals, perhaps via cron:

```
00,30 * * * * svn ci -m &ldquo;Regular scheduled check-in at: `date`&rdquo;
```

Your Apache configuration, as well as all previous revisions, are kept at a repository which may be viewed or checked out at any time.