By Fahd Aziz

Published: 2007-06-18 17:52

Setting Up A Subversion Repository Using Apache, With Auto Updatable Working Copy

Overview:

What is Subversion?

Subversion is a free/open-source version control system.That is, Subversion manages files and directories over time. A tree of files isplaced into a central repository. The repository is much like an ordinary fileserver, except that it remembers every change ever made to your files anddirectories. This allows you to recover older versions of your data, or examinethe history of how your data changed. In this regard, many people think of aversion control system as a sort of œtime machine•.

Subversion can access its repository across networks, whichallows it to be used by people on different computers. At some level, theability for various people to modify and manage the same set of data from theirrespective locations fosters collaboration. Progress can occur more quicklywithout a single conduit through which all modifications must occur. Andbecause the work is versioned, you need not fear that quality is the trade-offfor losing that conduit"if some incorrect change is made to the data, just undothat change.

Some version control systems are also software configurationmanagement (SCM) systems. These systems are specifically tailored to managetrees of source code, and have many features that are specific to softwaredevelopment"such as natively understanding programming languages, or supplyingtools for building software. Subversion, however, is not one of these systems.It is a general system that can be used to manage any collection of files. Foryou, those files might be source code"for others, anything from groceryshopping lists to digital video mixdowns and beyond.

Almost every Linux distribution comes with a standardsubversion installed.

The repository is of two formats bdb (berkeley db database) and fsfs (fsfsdatabase).

In our case we are using the FSFS database and therepository is created on /usr/local/subversion/repository

SVN has few methods to serve it's users. Below are someexamples:

1, SVN+SSH

2, SVN+Apache

3, SVNServe

In this case we are using the Apache method.

Apache should be running as an normal user, not nobody.

I won't guide people how to install apache in this how to.

Below is a step by step instruction on how to compilesubversion from the source code, and how to setup a repository using apachewebserver.

Documentation Contents:

Compiling subversion andits dependencies from source code:

First of all, we need get the source code for subversion andit's dependencies from **http://subversion.tigris.org/**

Some of the dependencies we need are,

APACHE (Webserver) (Source code isn't included in thesubversion dependencies.)

APR

APR-UTIL

NEON

After setting up the apache webserver, we need to compileAPR and APR-UTIL.

Do that by extracting the tarballs after downloading it from**http://subversion.tigris.org/**.

Extract both the subversion source code and the subversiondependencies source code.

```
tar
-jxvf subversion-x.x.x.tar.bz2
```

```
tar
-jxvf subversion-deps-x.x.x.tar.bz2
```

```
cd
subversion-x.x.x
```

Now we'll compile the APR first.

```
cd apr
```

```
./configure
--prefix=/usr/local/apr
```

```
make
```

```
make
install
```

```
cd ..
```

Next we'll compile APR-UTIL.

```
cd apr-util
```

```
./configure
--prefix=/usr/local/apr --with-apr=/usr/local/apr/
```

```
make
```

```
make
install
```

```
cd ..
```

After we're done with APR and APR-UTIL, we'll need tocompile NEON.

```
cd neon
```

```
./configure
--prefix=/usr/local/neon
```

```
make
```

```
make
install
```

```
cd ..
```

Finally we need to compile subversion with the support forall the we just installed.

```
./configure
--prefix=/usr/local/subversion --with-apxs={Location where you installed
apache}/bin/apxs --with-apr=/usr/local/apr/
--with-apr-util=/usr/local/apr-util/ --with-neon=/usr/local/neon/ --with-ssl
```

```
make
```

```
make
install
```

Creating a user for apache and modifying httpd.conf:

```
groupadd
apache
```

```
useradd
-g apache -d /usr/local/apache2
```

After installing apache we need to set ownership of all thefiles in `/usr/local/apache2` to user `apache`.

```
chown -Rv
apache.apache /usr/local/apache2
```

Finally we need to set which user the Apache server will berunning as.

Edit the default configuration file, or whatever configurationfile apache uses to run as.

I am going to assume the configuration file is `/usr/local/apache2/conf/httpd.conf.`

```
vi
/usr/local/apache2/conf/httpd.conf
```

Locate the line where it states something like.

```
User nobody
```

```
Group #-1
```

Make it look like this.

```
User apache
Group apache
```

Creating a repository:

Suppose I want to create a Repository at /usr/local/subversion/repositoryusing fsfs database so execute the command:

```
mkdir
-v /usr/local/subversion/
```

```
/usr/bin/svnadmin
create --fs-type fsfs /usr/local/subversion/repository
```

That should create a subversion repository under/usr/local/subversion/repository.

```
ls
/usr/local/subversion/repository
```

```
conf/  dav/ db/  format  hooks/ locks/  README.txt
```

You should be able to see those files under the repositorydirectory.

Setting up httpd.conf toserve the created repository:

Add the following lines to httpd.conf or the appropriateapache configuration file.

```
<Location /subversion>

  DAV svn

  SVNPath /usr/local/subversion/repository/

</Location>
```

Make sure that the module mod_dav is loaded in the apacheconfiguration file and is also present under modules directory.

Setting up authentication:

For the authentication we need to make changes to the apacheconfiguration yet another time.

Basic authentication requires that we just add the followinglines to the httpd.conf where we added the svn repository earlier.

```
AuthType Basic

AuthName "{Name of the authentication popup tab}"

AuthUserFile {Location of the password file}

Require valid-user
```

So it should look like this.

```
<Location /subversion>

  DAV svn
  SVNPath /usr/local/subversion/repository/

  AuthType Basic

  AuthName "Subversion repository"

  AuthUserFile /usr/local/subversion/repository/conf/svn-auth-file

  Require valid-user

</Location>
```

It is necessary that we add users to the password filebefore anyone can access it, which is described in the next step.

Adding SVN users:

Since we are using svn with an apache server, and an apachebasic authentication method.

We need to create a password file with the htpasswd binaryprovided with a standard apache installation.

```
htpasswd -cmd /usr/local/subversion/repository/conf/svn-auth-file
{user-name}
```

`-c` option creates a new htpasswd file.

`-m` encrypts the password with an MD5 algorithm.

`-d` encrypts the password with a CRYPT algorithm.

Where `{user-name}` stands for an actual user name that willbe used for authentication.

Warning: We should not use the `-c` option once we have addedthe first user. Using so will create and replace all existing user within thefile.

```
htpasswd
-md /usr/local/subversion/repository/conf/svn-auth-file {user-name}
```

Setting up the initialrepository layout:

A repository mostly contains 3 standard folders.

branches

tags

trunk

For creating those standard folders in a repository, createa temporary folder anywhere you want, /tmp would be a good idea, with thefollowing subdirectories.

```
mkdir
-pv /tmp/subversion-layout/{branches,tags}
```

After we have made all the layout folders, move all thecontents of your project to the trunk folder.

```
mv -v
/usr/local/apache2/htdocs /tmp/subversion-layout/trunk
```

Then make an initial import of the temporary createddirectory.

```
/usr/local/subversion/bin/svn
import /tmp/subversion-layout/ http://127.0.0.1/subversion/
```

This will setup you up with a default repository layout, andmake a first revision.

Setting up a working copy:

We can delete the temporary folders we created in the laststep, since all the files are already in the repository.

Now what we need to do is to make a working copy of all thefiles in the repository under */usr/local/apache2/htdocs*.

So that whenever a developer updates the php codes, they cansee the code changes taking effect in a working environment.

But setting up a working copy would not accomplish thistask, we would need to make the hook scripts to work with a working copy.

Thus, whenever a developer commits to the repository, thehook script will run itself, and update the working copy.

Make sure that htdocs folder under */usr/local/apache2/*doesn(TM)t already exist.

If you want you can rename it to *htdocs_old*.

To setup a working copy, do the following.

```
cd /usr/local/apache2/
```

```
su â€“ apache
```

```
/usr/local/subversion/bin/svn
checkout http://127.0.0.1/subversion/trunk/
htdocs
```

Setting up the hook scripts:

A hook is a program triggered by some repository event, suchas the creation of a new revision or the modification of an unversionedproperty. Each hook is handed enough information to tell what that event is,what target(s) it's operating on, and the username of the person who triggeredthe event. Depending on the hook's output or return status, the hook programmay continue the action, stop it, or suspend it in some way.

The hooks subdirectory is, by default, filled with templatesfor various repository hooks.

post-commit.tmpl	post-unlock.tmpl	pre-revprop-change.tmpl
post-lock.tmpl	pre-commit.tmpl	pre-unlock.tmpl
post-revprop-change.tmpl	pre-lock.tmpl	start-commit.tmpl

For now, I will be discussing about the *post-commit* hookscript, since that is what we need in our case.

Copy the *post-commit.tmpl* file into *post-commit* in the samehooks directory, and give *post-commit* execution rights.

```
cp -v
/usr/local/subversion/repository/hooks/post-commit.tmpl
/usr/local/subversion/repository/hooks/post-commit
```

```
chmod
+x /usr/local/subversion/repository/hooks/post-commit
```

Now edit the *post-commit* script and comment the follow twolines at the bottom, and add the following line to it.

```
#commit-email.pl "$REPOS" "$REV" commit-watchers@example.org

#log-commit.py --repository "$REPOS" --revision "$REV"

/usr/bin/svn update /usr/local/apache2/htdocs/ >> /usr/local/subversion/repository/logs/post-commit.log
```

After doing that, make a new folder logs, under `/usr/local/subversion/` so that we can enable logging, and create a blank the`post-commit.log` file.

```
mkdir
-v /usr/local/subversion/repository/logs/
```

```
touch
/usr/local/subversion/repository/logs/post-commit.log
```

Once again, we need to make sure the repository folder hasthe proper user ownership, it is advised to set ownership on `/usr/local/subversion/repository/` for user apache.

```
chown
-Rv apache.apache /usr/local/subversion/repository/
```

If all goes well, that's should be it.

You now have a working subversion repository server up whichis ready for further imports, as soon as you start the apache server.