

Setting up a CVS server on Linux, accessing via Windows over SSH

v1.0, May 13, 2004

Chris Pike - www.drydeadfish.co.uk

This document provides 'step-by-step' instructions for setting up a Concurrent Versions System (CVS) repository on Linux, and setting up WinCvs clients on Windows to access it over SSH (alternative access via Pserver is also briefly covered). It is designed to help get everything up and running, not as a replacement for the CVS manual (from which some of this documentation was taken).

This document assumes that both the SSH and CVS servers are installed on your Linux box; if not, just install from your Linux distro CD, or download and install the latest versions:

CVS info and downloads: <http://www.cvshome.org>

SSH info and downloads: <http://www.openssh.com>

RPM downloads from: <http://www.rpmfind.net>

Contents

Linux

1. [Enable the services](#)
2. [Creating the repository](#)
3. [Adding code to the repository](#)
4. [Accessing code within the repository](#)

Windows

1. [Installing the components](#)
2. [Setting up SSH access](#)
3. [WinCvs settings](#)
4. [Checking out a module](#)
5. [Pserver - if you're desperate to use this method](#)

Linux

Enable the services

Before doing anything else, make sure that CVS and SSH are there waiting for a connection:

- 1) Log in as root.

```
$ su root
```

- 2) Start CVS service, and ensure that it starts on system reboot.

```
# service cvs start
```

```
# chkconfig cvs on
```

3) Start SSH service, likewise.

```
# service sshd start  
# chkconfig sshd on
```

Again, if you don't have either of these services installed, they should be available as packages with your Linux distro. You can also download them from the Websites (listed at the beginning of this document).

Creating the repository

A group will need to be set up for the CVS users, the repository will need to be created and permissions set so that group users can access it.

The directory for the repository can be named whatever you like (but it must not contain spaces), it's usually referred to as 'cvsroot'. The location can be wherever you like, but here it's going to be located at '/usr/local/cvsroot' (that's where it is in the manual, so that's where I'll put it).

The following steps will set up the repository:

1) Set the environment variables.

To enable the following variables for all users, add the following to the file named 'profile' which is located in the '/etc' directory:

```
export CVSROOT=/usr/local/cvsroot
```

If you have users accessing the CVS repository via the command line, you may also wish to set the text editor variable:

```
export CVSEDITOR=/bin/vi
```

2) Create the group for CVS users.

You will need to set up a group for all CVS users, so that they have group permission to access the repository.

Log in as root and first of all make the variable \$CVSROOT immediately available (the variable you added to '/etc/profile' will only be set after you log out and log back in; therefore, unless you've done so, you must specify it manually):

```
$ su root  
# export CVSROOT=/usr/local/cvsroot
```

Next, create a new user group (called 'cvs' in this case):

```
# groupadd cvs
```

3) Adding new users to the 'cvs' group.

Anybody that wishes to access the CVS server will need to have a local account on the server itself. If you need to add new users to the Linux server, do the following:

```
# useradd -g cvs -m <username>
```

For example, to add user 'george', do the following:

```
# useradd -g cvs -m george
```

This will add the new user and will also add the user to the 'cvs' group.

4) Adding existing users to the 'cvs' group.

Any users already on the Linux server that require CVS access will need to be added to the CVS group.

To find out what group(s) a particular user is already part of, you can use the 'groups' command, followed by the user name:

```
# groups fred  
fred : fred staff admin
```

The above example shows user fred as belonging to groups 'fred', 'staff' and 'admin'.

If user 'fred' is not part of any existing groups, you would do the following:

```
# usermod -G cvs fred
```

If user 'fred' is currently part of groups 'staff' and 'admin', do the following:

```
# usermod -G cvs,staff,admin fred
```

Note: groups are separated by commas, without spaces!

When adding a user to the 'cvs' group, if you don't specify any groups the user currently belongs to, the user will be removed from these groups. A user will not be removed from a group that is the same as their user name, so you don't need to specify this.

5) Create the directory for the CVS repository.

```
# mkdir $CVSROOT
```

6) Set permissions for the repository directory.

Change the group ownership and permission of \$CVSROOT (and all files within it) to group 'cvs':

```
# chgrp cvs $CVSROOT  
# chmod 770 $CVSROOT  
# chmod g+s $CVSROOT
```

Note: 'chmod g+s' sets the 'Group ID' (GID) for the directory, meaning that any new files/directories that are created in the directory will be of the same group as the directory itself.

7) Initialize the CVS repository.

This will set up the directory you specified in \$CVSROOT as the repository, generating the necessary files and directories required:

```
# cvs init
# exit
```

Adding code to the repository (on the Linux server)

To check that everything is set up correctly, it's a good idea to add some code to the repository.

1) Go to the directory currently containing your source code.

First, make sure that you're logged in as a user of the 'cvs' group:

```
$ su <user>
```

Note: Do this, even if you're already logged in as this user, this will ensure that you have group permission to access the CVS repository.

Next, make the variable \$CVSROOT available, then change to the directory where your code is currently located, for example:

```
$ export CVSROOT=/usr/local/cvsroot
$ cd ~/mydir/myapp/src
```

Note: the export command is only required if you haven't logged out and logged back in after updating your '/etc/profile' file earlier.

2) Import the code to the CVS repository.

In the following example, we'll assume that we want the code imported to \$CVSROOT/gibletpsoft/text_editor/

```
$ cvs import -m "Imported sources" gibletpsoft/text_editor Gibletpsoft V1_0_0
```

The command to import the source files is explained as follows:

```
$ cvs import -m <"log message"> <path to save code to> <vendor> <revision>
```

- `-m <"log message">`
The quoted text after the '-m' flag is a log message, you can type what you like here. If this is not specified then CVS will start your editor and prompt for a message (which is annoying).
- `<path to save code to>`
After this you should specify the path within the repository that you want your code stored in. The directories you specify will be automatically created within the repository.
- `<vendor> <revision>`
These tags are required by CVS. For more details on this check out the CVS manual.

Accessing code within the repository

To check that it's all working, seeing that we've added source code to the repository, it would be fitting to define the module. Modules are not necessary, but are convenient in grouping together related files and directories (so the manual says, and I believe it).

1) Create a temporary directory.

First, make sure you're logged in as a user of group 'cvs':

```
$ su <user>
```

The following creates a temporary directory named 'mycvmdir' in the home directory of the user you're logged in as, and then enters that directory:

```
$ mkdir ~/mycvmdir  
$ cd ~/mycvmdir
```

2) Copy the 'modules' file from the repository.

The 'checkout' command below will get a working copy of the 'modules' file from the \$CVSROOT/CVSROOT directory and place it in the directory you're currently in:

```
$ cvs checkout CVSROOT/modules
```

The directory structure of the file you checked out is also copied to your current directory, so you should now have the directory structure '~/mycvmdir/CVSROOT' which contains the 'modules' file.

3) Editing the 'modules' file.

Enter the CVSROOT directory:

```
$ cd CVSROOT
```

Edit the 'modules' file within this directory, and add a definition for the module at the end of the file like this:

```
text_editor gibletfoot/text_editor
```

First the module name is defined as 'text_editor', then the path to the code is defined as 'gibletfoot/text_editor'. Save the file when done.

4) Commit the updated 'modules' file.

```
$ cvs commit -m "Added text_editor module" modules
```

That's it. The 'modules' file in \$CVSROOT/CVSROOT should now be updated with the definition of your module.

Note: if you don't specify a comment (via the -m option) then a text editor will be invoked so that you can specify it (highly annoying, as I mentioned previously).

5) Remove the working directory.

To delete the working directory and its contents, just type the following (unless you wish to keep it for nostalgic reasons):

```
$ cd..
```

This takes you out of the CVSROOT directory and back to the parent directory (~/.mycvsdir). You can't release the directory if you're currently in it, so the above command is necessary!

```
$ cvs release -d CVSROOT
```

When prompted, type 'y', then hit the 'enter' key.

The 'release' command checks that no uncommitted changes are present, you don't really need to use it in this instance, but there you go.

Windows

Installing the components

You will need to download the following applications to access the CVS repository via the Windows client:

WinCVS - www.wincvs.org

Python - www.python.org

OpenSSH for Windows - sshtools.sourceforge.net

This document was tested with the following versions:

WinCvs 1.3.13.2 Beta 13 (build 2)

Python 2.3.3

OpenSSH for Windows 3.7.1p1-1

1) Install WinCvs.

Extract 'setup.exe' from the zip file and run it.

2) Install Python.

Just run installation file (no configuration necessary).

Note: You will need to have Python installed for WinCvs to work.

3) Install OpenSSH for Windows.

Extract 'setup.exe' from the zip file and run it.

When prompted for the location to install SSH to, change this to 'C:\ssh' (by default it will be 'C:\Program Files\OpenSSH').

Setting up SSH access

The following will set up SSH access for the Windows machine, and create private and public keys, of which the public key will be uploaded to the Linux server to allow access without requiring password authentication.

Note: Although this method is efficient, if anybody cracks your machine and obtains your RSA keys, they will have access to the SSH server! You have been warned!

1) Make sure you have the following information.

- Name or IP address of the Linux server
- Username and password for logging on to the Linux server
- Path to \$CVSROOT on the Linux server

2) Create your HOME directory.

Make a new directory in your C:\ drive named 'Home' (i.e. 'C:\Home').

3) Set the environment variables HOME and PATH.

Edit your 'C:\Autoexec.bat' file and add the following to the end:

```
SET HOME=C:\Home  
SET PATH=%PATH%;C:\ssh\bin
```

4) Make an SSH connection.

Open up a command prompt, and first make the environment variables immediately available by typing the following:

```
> SET HOME=C:\Home  
> SET PATH=%PATH%;C:\ssh\bin
```

then run the SSH client and log onto the server using your user name and the domain (or IP address) of the server:

```
> ssh <user>@<remote host>
```

Enter the password when asked, this will log you onto the SSH server.

At this point, in your local %HOME% directory on the Windows machine, a directory named '.ssh' will automatically be created along with the files 'known_hosts' and 'random_seed'.

5) Set up Linux user account for password-less SSH access.

While you're logged onto the server make sure the ~/.ssh directory exists:

```
$ ls -al
```

This should list all files and directories. If '.ssh' is not listed, do the following:

```
$ mkdir ~/.ssh
```

To ensure that the directory has the correct permissions, set the permissions to "700":

```
$ chmod 700 .ssh
```

When you're sure everything is present and correct, log out:

```
$ exit
```

6) Create a key pair.

Type the following:

```
> cd %HOME%\ssh  
> ssh-keygen -t rsa -f identity -N "" -C <comment>
```

This will generate a key with a null passphrase.

Note: For <comment>, basically enter a comment to help identify the key, if this is omitted it would be 'user@host' (where 'user' is your user name on the local machine, and 'host' is the domain name of the local machine). You can specify something like fred@domain.com or "fred's key" if you feel like it. If you specify a comment that contains spaces, you must enclose it in quote marks!

This will create two files in the '%HOME%\ssh' directory named 'identity' (your private key) and 'identity.pub' (your public key).

7) Copy your public key to the server.

Copy the identity.pub key to your home directory on the Linux server by doing the following:

```
> scp identity.pub <user>@<remote host>:~/identity.pub
```

8) Log into the server and add the key to the 'authorized_keys' file.

```
> ssh <user>@<remote host>
```

Once you're logged in, type the following:

```
$ cat identity.pub >> .ssh/authorized_keys  
$ rm identity.pub
```

This will add the contents of 'identity.pub' to the authorized_keys file (if the authorized_keys file doesn't exist then it will create it).

NOTE: the 'authorized_keys' file must have permissions set to "600". If you're not sure, type the following:

```
$ chmod 600 ~/.ssh/authorized_keys  
$ exit
```

9) Check that access via public key is successful.

Try logging in again:

```
> ssh <user>@<remote host>
```

You should be logged in without being asked for a password.

Note: If you are asked for a password, go back and make sure you followed all of the above steps -

especially check that you've set the permissions correctly.

WinCvs settings

Run the WinCvs application, and configure the settings as follows:

1) Click on Admin | Preferences

2) On the General tab, set the following:

- **Authentication:** 'ssh' - select this option from the drop-down menu, then click on the 'settings' button and specify the path to SSH (C:\ssh\bin\ssh.exe).

NOTE: If you use 'pserver' as an authentication method here, you don't need to set up a key for SSH, instead you log in manually each time via WinCvs. This method does not use encryption, and is not recommended. More information on how to use Pserver can be found at the end of this document.

- **Path:** '/usr/local/cvsroot/' - or wherever \$CVSROOT is on the server
- **Host address:** '192.168.1.2' - or wherever the server's IP or domain is
- **User name:** 'fred' - or the user on the Linux server you'll log in as
- **CVSROOT:** (this is updated automatically from the above info)

3) On the Globals tab, check the following boxes:

- Supply control when adding files
- Quiet mode
- Dirty files support
- Prune empty directories
- Disable splash screen

4) On the CVS tab, set the following:

- **Home:** 'C:\Home' (or whatever you set your HOME directory to)

5) Click OK.

Stop and restart WinCvs after making these settings.

Checking out a module

To verify that everything works, check out the test module that you set up in the repository:

1) Click on Remote | Checkout Module

2) Specify the Module name and path (e.g. 'gibletoft/text_editor')

3) Specify your local directory to check the files out to (e.g. 'C:\cvs'). If the directory doesn't currently exist then it will ask you if it should create it, and if you click 'Yes' it will do so (which is rather thoughtful of it).

4) Click OK.

You should see the source code files from '\$CVSROOT/gibletoft/text_editor' now displayed in the right-hand pane of the window. If you get an error, check your WinCvs settings.

Pserver

Accessing the CVS server via Pserver does not use encryption.

If you use Pserver as an authentication method, you don't need to set up SSH. To access the CVS server you log in via WinCvs by doing the following:

- 1) Click on Admin | Login.
- 2) Enter your password when prompted.

Note: You will need to log in manually each time, entering your user password on the Linux server.

You will also need to do the following on the Linux server:

Edit 'etc/cvs/cvs.conf', and add the following line to specify the path of your CVS repository:

```
CVS_REPO="/usr/local/cvsroot"
```