

CVS & Subversion

Revision control

Revision control (also known as version control, source control or (source) code management (SCM)) is the management of multiple revisions of the same unit of information. It is most commonly used in engineering and software development to manage ongoing development of digital documents like application source code, art resources such as blueprints or electronic models and other critical information that may be worked on by a team of people. Changes to these documents are identified by incrementing an associated number or letter code, termed the "revision number", "revision level", or simply "revision" and associated historically with the person making the change. A simple form of revision control, for example, has the initial issue of a drawing assigned the revision number "1". When the first change is made, the revision number is incremented to "2" and so on.

The most popular revision control is CVS and Subversion.

CVS

The Concurrent Versions System (CVS), also known as the Concurrent Versioning System, is an open-source version control system invented and developed by Dick Grune in the 1980s. CVS keeps track of all work and all changes in a set of files, typically the implementation of a software project, and allows several (potentially widely-separated) developers to collaborate. CVS has become popular in the open source software world and is released under the GNU General Public License.

Subversion

Subversion is a revision control system which allows computer software to be developed in an incremental and controlled fashion by a distributed group of programmers. Also commonly referred to as svn or SVN, Subversion is designed specifically to be a modern replacement for CVS. Subversion was created by CollabNet, who still maintain the project. The name "Subversion" is a trademark of CollabNet.

Although as of 2006 Subversion is less widely used than the traditional CVS, adoption is increasing, and it is perhaps the most popular alternative.[citation needed] Projects using Subversion include the Apache Software Foundation, KDE, GNOME, GCC, Python, Samba, Mono and many others. SourceForge.net now also provides Subversion hosting for its open source projects, and the new Google Code and BountySource systems use it exclusively.

The latest Subversion release is 1.4.3, released on 25 January 2007. Released under the Subversion License, Subversion is open source software.

Features CVS

CVS uses client-server architecture: a server stores the current version(s) of the project and its history, and clients connect to the server in order to check out a complete copy of the project, work on this copy and then later check in their changes. Typically, client and server connect over a LAN or over the Internet, but client and server may both run on the same machine if CVS has the task of keeping track of the version history of a project with only local developers. The server software normally runs on Unix (although at least CVSNT server supports various flavors of Windows and Unix), while CVS clients may run on any major operating-system platform.

Several developers may work on the same project concurrently, each one editing files within their own working copy of the project, and sending (or checking in) their modifications to the server. To avoid the possibility of people stepping on each other's toes, the server will only accept changes made to the most recent version of a file. Developers are therefore expected to keep their working copy up-to-date by incorporating other people's changes on a regular basis. This task is mostly handled automatically by the CVS client, requiring manual intervention only when a conflict arises between a checked-in modification and the yet-

unchecked local version of a file.

If the check-in operation succeeds, then the version numbers of all files involved automatically increment, and the CVS server writes a user-supplied description line, the date and the author's name to its log files. CVS can also run external, user-specified log processing scripts following each commit. These scripts are installed by an entry in CVS's loginfo file, which can trigger email notification, or convert the log data into a Web-based format.

Clients can also compare algorithms, request a complete history of changes, or check out a historical snapshot of the project as of a given date or as of a revision number. Many open-source projects allow "anonymous read access", a feature that was pioneered by OpenBSD. This means that clients may check out and compare versions with either a blank or simple published password (e.g. "anoncvs"); only the check-in of changes requires a personal account and password in these scenarios.

Clients can also use the "update" command in order to bring their local copies up-to-date with the newest version on the server. This eliminates the need for repeated downloading of the whole project.

CVS can also maintain different "branches" of a project. For instance, a released version of the software project may form one branch, used for bug fixes, while a version under current development, with major changes and new features, forms a separate branch.

CVS uses delta compression for efficient storage of different versions of the same file. The implementation favors files with many lines (usually text files) - in extreme cases individual copies of each version are stored rather than a delta.

Features Subversion

- * Commits are truly atomic operations. Interrupted commit operations do not cause repository inconsistency or corruption.
- * Renamed/copied/moved/removed files retain full revision history.
- * Directories, renames, and file metadata are versioned. Entire directory trees can be moved around and/or copied very quickly, and retain full revision history.
- * Versioning of symbolic links.
- * Native support for binary files, with space-efficient binary-diff storage.
- * Apache HTTP server as network server, WebDAV/DeltaV for protocol. There is also an independent server process that uses a custom protocol over TCP/IP.
- * Branching and tagging are cheap (constant time) operations.
- * Natively client/server, layered library design.
- * Client/server protocol sends diffs in both directions.
- * Costs are proportional to change size, not data size.
- * Parsable output, including XML log output.
- * Open Source licensed — "CollabNet/Tigris.org Apache-style license"
- * Internationalised program messages.
- * File locking for unmergeable files ("reserved checkouts").
- * Path-based authorization for svnserve.
- * Python, Ruby, Perl, and Java language bindings.
- * Full MIME support - the MIME Type of each file can be viewed or changed, with the software knowing which MIME types can have their differences from previous versions shown.

Common vocabulary

Repository

The repository is where the current and historical file data is stored, often on a server. Sometimes also called a depot (e.g. with SVK, AccuRev and Perforce).

Working copy

The working copy is the local copy of files from a repository, at a specific time or revision. All work done to the files in a repository is initially done on a working copy, hence the name. Conceptually, it is a sandbox.

Check-out

A check-out (or checkout or co) creates a local working copy from the repository. Either a specific revision is specified, or the latest is obtained.

Commit

A commit (check-in, ci or, more rarely, install or submit) occurs when a copy of the changes made to the working copy is written or merged into the repository.

Change

A change (or diff, or delta) represents a specific modification to a document under version control. The granularity of the modification considered a change varies between version control systems.

Change list

On many version control systems with atomic multi-change commits, a changelist, change set, or patch identifies the set of changes made in a single commit. This can also represent a sequential view of the source code, allowing source to be examined as of any particular changelist ID.

Update

An update (or sync) merges changes that have been made in the repository (e.g. by other people) into the local working copy.

Branch

A set of files under version control may be branched or forked at a point in time so that, from that time forward, two copies of those files may be developed at different speeds or in different ways independently of the other.

Merge

A merge or integration brings together two sets of changes to a file or set of files into a unified revision of that file or files.

* This may happen when one user, working on those files, updates their working copy with changes made, and checked into the repository, by other users. Conversely, this same process may happen in the repository when a user tries to check-in their changes.

* It may happen after a set of files has been branched, then a problem that existed before the branching is fixed in one branch and this fix needs merging into the other.

* It may happen after files have been branched, developed independently for a while and then are required to be merged back into a single unified trunk.

Dynamic stream

A stream (a data structure that implements a configuration of the elements in a particular repository) whose configuration changes over time, with new versions promoted from child workspaces and/or from other dynamic streams. It also inherits versions from its parent stream.

Revision

A revision or version is one version in a chain of changes.

Tag

A tag or release refers to an important snapshot in time, consistent across many files. These files at that point may all be tagged with a user-friendly, meaningful name or revision number.

Import

An import is the action of copying a local directory tree (that is not currently a working copy) into the repository for the first time.

Export

An export is similar to a check-out except that it creates a clean directory tree without the version control metadata used in a working copy. Often used prior to publishing the contents.

Conflict

A conflict occurs when two changes are made by different parties to the same document, and the system is unable to reconcile the changes. A user must resolve the conflict by combining the changes, or by selecting one change in favour of the other.

Resolve

The act of user intervention to address a conflict between different changes to the same document.

Baseline

An approved revision of a document or source file from which subsequent changes can be made.

Reverse integration

The process of merging different team branches into the main trunk of the versioning

system.

You can install CVS from <http://ftp.gnu.org/non-gnu/cvs/source/stable/1.11.22/>

The Basics

CVS and Subversion both maintain a central “repository”, which is a place where they store all the versions of your project that you want under revision control.

The details of how they work is very different but the basic ideas behind repositories and working copies are remain the same for CVS and Subversion:

- * The repository is kept in a central place like on a server.
- * When you “check out” code, CVS/Subversion fetches a “working copy” of the project from the repository and stores the copy on your computer.
- * You can edit this working copy all you like, or even delete it. The server makes no attempt to keep track of checked out copies and won't care what you do with your copy unless you try to commit your changes.
- * The checked out copies do keep track of themselves and where you got them from. They do this using additional sub-directories that CVS or Subversion add to your projects directories.
- * When you perform a “commit” operation, CVS/Subversion detects which files you have changed and sends your changes back to the server.

CVS is self contained in one executable program, called “cvs”. You tell it what to do by providing different options, for example “cvs import”, or “cvs commit”. Subversion is also fairly self contained but has two executables: “svn” and “svnadmin”. The “svnadmin” program has all the commands for creating and directly modifying repositories. The “svn” program is purely a client that handles all the day to day operations on an existing repository. Two typical operations are “svnadmin create” and “svn commit”.

The Repository

The repository is the main database where CVS / Subversion stores the “official copy” of your project and all previous revisions.

These examples create CVS / Subversion repositories in your home directory. Both CVS and Subversion offer mechanisms to store the repository on a remote server but we'll keep it simple for now and you can look up how to set up servers once you're confident.

Before you start, put together a “test project” to import into the repository. Use an existing project if you like (but don't trust your only copy of something to a tutorial!). I'll assume for the sake of the examples that you created a directory called “myproject” containing three text files: A.txt B.txt and C.txt, each containing about eight lines of text. It should be easy enough to adapt the examples to the actual files you chose.

To create a blank repository and then “import” your project to it:

CVS:

```
cvs -d /home/cvs/cvsrepos init
  cd myproject
  cvs -d /home/cvs/cvsrepos import -m "Imported my project at version 1.0" mycvsproject
MYVENDOR VER_1_0
```

Subversion:

```
svnadmin create /home/svn/svnrepos
  cd myproject
  svn import . file:///home/svn/svnrepos/trunk/mysvnproject --message 'Imported my project
at version 1.0'
```

Configuring CVS

The following steps give simple steps to install CVS software

- 1) Download the tarball cvs-1.11.1p1.tar.gz from <http://www.cvshome.com>

- 2) Run "gunzip cvs-1.11.1p1.tar". You will get file called cvs-1.11.1p1.tar.

- 3) Then extract this file using command

```
"tar -xvf cvs-1.11.1p1.tar".
```

- 4) Then go to directory cvs-1.11.1p1.

- 5) There you will find a file called INSTALL. Read it once carefully.

- 6) Now run the command "./configure."

- 7) Build it using the command "make".

- 8) Install binaries/documentation using command "make install".

- 9) Now open file called /etc/services and add the following lines.

```
Cvpsserver 2401/tcp # CVS client/server operations
```

```
cvpsserver 2401/udp # CVS client/server operations
```

- 10) Open file called /etc/xinetd.conf and add the following lines to that file.

```
service cvpsserver
```

```
{
```

```
Port = 2401
```

```
socket_type = stream
```

```
protocol = tcp
```

```
wait = no
```

```
user = root
```

```
passenv =
```

```
server = /opt/bin/cvs
```

```
server_args = --allow-root=/home/cvs -f pserver
```

```
env = HOME=/home/cvs
```

```
log = /var/log/cvslog
```

```
}
```

- 11) Space on both sides of = in the above file is a must.

- 12) Restart xinetd services by running command "services xinetd restart"

- 13) Check whether CVS is running on the port 2401 by using the following command "nmap 192.168.16.17 |grep 2401". If the CVS server is running correctly it will give output as "2401/tcp open cvpsserver".

- 14) Suppose you want to put my CVS repository at /home/cvs. Then create an environmental variable called CVSROOT which points to /home/cvs.

- 15) Create the actual repository by the command "cvs -d /home/cvs init".

- 16) Create a link to /etc/password to file called passwd.

- 17) Now go to the client machine and install the Wincvs available from //titan/software/cvs.

- 18) Open Wincvs window. Go to admin>>preferences.

- 19) At the 'Enter CVSROOT' field enter ":pserver:username@cvsserver_name:/home/cvs".

- 20) In the Authentication field select "password file on cvs server".

- 21) Now it is time to create to new modules. In Wincvs client go to Create>>Import module. Select the folder from which you want to create the module.

- 22) Then in "import settings" dialog box enter the module name as whatever you want.

Mention vendor tag and release tag appropriately. You can also give log message that is associated with this module.

- 23) Now any other client machine access the same modules

- 24) To access the modules from other machines, open wincvs client in that machine. Repeat steps 18 and 19.

- 25) Now click on Admin>>login. It will ask for the password. Supply the correct password.

- 26) Then go to Create checkout module. In "Enter the module name field give the module name you created in the above steps"

- 27) In the field "local folder to checkout to" give the path of the folder wherever you want to put your files.

- 28) Start using CVS as usual.

Setting Up Subversion

The subversion versioning system is great and comes with many features and documentation. This text will focus on getting, configuring and using a subversion repository quickly.

Getting, Configuring and Building Subversion

In the example, subversion is installed to a single tree to simplify things. The sources are available for download from Tigris'. The initial steps are unpack then configure.

```
tar xjvf subversion-1.3.1.tar.bz2
cd subversion-1.3.1
./configure --prefix=/usr/local/subversion
make
```

(as root or sudo)

```
make && make install
```

Limitations CVS

For each commonly listed limitation of CVS there is also a commonly listed reason:

- * Moving or renaming of files and directories are not versioned. Refactoring should be avoided in a development process and can be managed by an administrator (by moving the RCS file) when it is (rarely) required. If you develop in Oracle Forms, Cobol, Fortran or even C++ then the CVS reasoning is quite commonly accepted; if you develop with Java then the CVS reasoning may seem counterintuitive.

- * No versioning of symbolic links. Symbolic links stored in a version control system can be a security risk - someone can create a symbolic link index.htm to /etc/passwd and then store it in the repository; when the "code" is exported to a Web server the Web site now has a copy of the system security file available for public inspection. A developer may prefer the convenience and accept the responsibility to decide what is safe to version and what is not; a project manager or auditor may prefer to reduce the risk by using build scripts that require certain privileges and conscious intervention to execute.

- * Limited support for Unicode text files and non-ASCII filenames. Unix systems run in UTF-8 and so CVS on Unix handles UTF-8 filenames and files natively. If you only work on Unix systems then this response seems reasonable; however when you work on AS/400 and Windows it may not.

Over time, developers have wanted to change the CVS code significantly to add new features, refactor the code and improve developer productivity. This has led to the phrase YACC: "Yet Another CVS Clone". CVS replacement projects include CVSNT (first released 1998), OpenCVS (to be released soon) and Subversion (first released 2004).

Limitations Subversion

A number of issues with Subversion have been identified by the developers as well as users.

- * No atomic 'move' or 'rename' operation. As of version 1.4, Subversion currently implements moving (or renaming) of files with two operations on the same atomic transaction: a 'copy' to the new name followed by a 'delete' of the old name.

- * Subversion currently lacks proper repository administration and management tools. That is, while it is very capable when data is added to the repository, it is much less capable at managing the repository as a whole. For instance, it is sometimes necessary to make permanent edits to the repository to change the structure in which versions are held or to permanently remove data that was checked into the repository in error. Subversion does not have tools which allow this to be done. The check-in level tools allow files and directories to be moved or deleted but earlier revisions will always hold the data in the old structure or hold the file that was deleted. The current solution to this sort of problem involves 'dumping' the repository, editing the resulting (possibly large) text file, and then recreating the repository. For simple renaming or removal of files this is fairly straight-forward, but other alterations can be more complex and hence error-prone.

Reference

- 1, <http://www.nongnu.org/cvs/>
- 2, http://en.wikipedia.org/wiki/Concurrent_Versions_System
- 3, <http://subversion.tigris.org/>
- 4, [http://en.wikipedia.org/wiki/Subversion_\(software\)](http://en.wikipedia.org/wiki/Subversion_(software))
- 5, <http://www.developingprogrammers.com/index.php/2005/11/24/cvs-and-subversion-combined-tutorial/>
- 6, http://en.wikipedia.org/wiki/Revision_control
- 7, <http://www.freeos.com/articles/4608/>
- 8, <http://systhread.net/texts/200607subver.php>