Subversion - svn

Source Code Management System

Dinesh Patil

Agenda

- Why move to Subversion?
- Subversion client side
- Subversion Installation
- Basic Work Cycle
- How do Revisions Work?
- Conflict Resolution
- Tags/branches
- Log Messages
- Subversion local repository
- Subversion through Netbeans

Why move to Subversion? (CVS vs SVN)

- Functional replacement for CVS
- Directory versioning (renames and moves)
- Atomic commits (all or nothing)
- Faster network access (binary diffs)
- File & directory meta-data
- Requires less network access

Subversion Client Side

- Each working directory has a .svn directory
- Similar to CVS's CVS directory
- Repository password stored in $HOME/.svn
- Stores a pristine copy of each file in directory

Subversion Installation

- Windows, Linux, Mac : Easy installation
  - Refer to instructions here: http://downloads.open.collab.net/collabnet-subversion.html
- Ubuntu: "sudo apt-get install subversion"
- Solaris 9, 10
  - Install all dependent packages and subversion from: http://www.sunfreeware.com/
  - Easier option is to follow these instructions.

    Solaris 10: http://www.blastwave.org/howto.html

    Solaris 8/9:http://www.blastwave.org/howto_S8.html

    Last step : To install : # pkg-get -i subversion

SVN Command Line Differences

- CVS
  - Argument position matters

```
% cvs -d /export/home1/cvs update -d
```

- SVN
  - Argument position does not matter

```
% svn log -r 123 foo.c
% svn log foo.c -r 123
```

  - Authenticating the server

```
$ svn command URL…
Password for 'user':  XXXXXX
```

Basic Work Cycle (Part 1)

- Checkout a working copy
- Update working copy
- Make changes
- Examine your changes
- Merge other's changes
- Commit your changes

Basic Work Cycle (Part 2)

- Checkout a working copy

```
% svn checkout file:///c:/sun/glassfish-svn
```

- Update working copy
  - Update all files and directories to the most current version

```
% svn update
```

  - Go to a particular older revision for all files and directories

```
% svn update -r 1345
```

  - I want an even older version of svn-doc.el

```
% svn update -r 999 svn-doc.el
```

Basic Work Cycle (Part 2)

- Update output
  - U `foo' : File `foo' was (U)pdated received changes from the server.)
  - A `foo' : File or directory `foo' was (A)dded to your working copy.
  - D `foo' : File or directory `foo' was (D)eleted from your working copy.
  - R `foo': File or directory `foo' was (R)eplaced in your working copy; that is, `foo' was deleted, and a new item with the same name was added. While they may have the same name, the repository considers them to be distinct objects with distinct histories.
  - G `foo' : File `foo' received new changes, but also had changes of your own to begin with. The changes did not intersect, however, so Subversion has mer(G)ed the repository's changes into the file without a problem.
  - C `foo' : File `foo' received (C)onflicting changes from the server. The changes from the server directly overlap your own changes to the file. No need to panic, though. This overlap needs to be resolved by a human (you).

Basic Work Cycle (Part 3)

- Make changes
  - Add new files and directories

```
% vi blair_super_algorithm.c
% mkdir data-files
% touch data-files/file1
% svn add blair_super_algorithm.c data-files
# data-files/file1 added unless -N passed to svn add
```

  - Delete files

```
% svn rm foo what_do_they_know.c
```

  - Rename file

```
% svn mv README.txt README_OLD.txt
# This is identical to
% svn cp README.txt README_OLD.txt; svn rm README.txt
```

  - Copy files and directories

```
% svn cp test_datafiles test_datafiles_new
# If test_datafiles is a directory, then test_datafiles_new
# is an exact copy of test_datafiles
```

Basic Work Cycle (Part 4)

- Examine your changes : svn status: Normal amount of information

```
% svn status

_ L     ./abc.c                [svn has a lock in its .svn directory for abc.c]

M       ./bar.c                [the content in bar.c has local modifications]

_M      ./baz.c                [baz.c has property but no content modifications]

?       ./foo.o                [svn doesn't manage foo.o]

!       ./foo.c                [svn knows foo.c but a non-svn program deleted it]

~       ./qux                  [versioned as dir, but is file, or vice versa]

A  +    ./moved_dir            [added with history of where it came from]

M  +    ./moved_dir/README     [added with history and has local modifications]

D       ./stuff/fish.c         [this file is scheduled for deletion]

A       ./stuff/things/bloo.h [this file is scheduled for addition]
```

Basic Work Cycle (Part 5)

- Examine your changes
  - svn status: More information with -v
    - First column the same
    - Second column, working revision
    - Third column, last changed revision
    - Fourth column, who changed it

```
% svn status -v
M               44        23    joe       ./README
_               44        30    frank     ./INSTALL
M               44        20    frank     ./bar.c
_               44        18    joe       ./stuff
_               44        35    mary      ./stuff/trout.c
D               44        19    frank     ./stuff/fish.c
_               44        21    mary      ./stuff/things
A                0         ?     ?        ./stuff/things/bloo.h
_               44        36    joe       ./stuff/things/gloo.c
```

Basic Work Cycle (Part 6)

- Examine your changes
  - svn status: Even more information with -u
    - Asterisks show if the file would be updated if `svn update` were run
    - Requires network access to the repository

```
% svn status -u -v
M      *         44         23    joe        ./README
M                44         20    frank      ./bar.c
_      *         44         35    mary       ./stuff/trout.c
D                44         19    frank      ./stuff/fish.c
A                 0          ?     ?         ./stuff/things/bloo.h
```

Basic Work Cycle (Part 7)

- Examine your changes
  - svn diff: Show your modifications
    - Even shows modifications in properties
    - Show all differences in files and directories in local working copy

```
% svn diff
```

    - Diff between revision 3 of foo.c in repository and local working foo.c

```
% svn diff -r 3 foo.c
```

    - Diff between revisions 2 and 3 of foo.c in the repository without even touching local foo.c

```
% svn diff -r 2:3 foo.c
```

- Revert or undo your changes : "svn revert" can cancel the "svn add" and "svn rm"
  - Does not require network access

```
% svn revert
```

- Commit your changes

```
% svn commit -m "Add message"
```

How do Revisions work? (Part 1)

- Revision numbers are applied to an object to identify a unique version of that object
- Example files

```
% ls

bar.c foo.c
```

- CVS
  - Revision numbers are per file
  - A revision number from one file does not necessarily have any meaning to another file with the same revision number
  - A commit only updates the revision numbers of files that were modified
  - bar.c may be at revision 1.2 and foo.c at 1.10
  - Updates to bar.c will not change foo.c's revision number

How do Revisions work? (Part 2)

- Subversion
  - Revision numbers are global across the whole repository
  - Identify how the entire repository looks at that instant in time
  - A commit creates a snapshot of the entire tree in the repository at that revision number
  - Allows users to say, “Hey so-and-so, go get revision 1432 of XYZ and try to compile it.”
  - Before an update, both bar.c and foo.c are at revision 25
  - Modify bar.c and commit
  - Then update the working copy
  - Now bar.c and foo.c are at revision 26, except that foo.c in revision 25 and 26 are identical
  - No additional space in repository required, i.e. a cheap copy or a symbolic link is made

How do Revisions work? (Part 3)

- Example revision update
  - Check out tree

```
% svn co http://svn.somewhere.com/repos
A  Makefile
A  document.c
A  search.c
Checked out revision 4.
```

  - Edit search.c

```
% vi search.c
```

  - Commit the edit

```
% svn commit –m “Add better search”
Sending search.c
Transmitting data…
Committed revision 5.
```

How do Revisions work? (Part 4)

- Status of files and directories
  - Directory `.' is at 4 but revision 5 exists in the repository
  - Makefile is at 4
  - document.c is at 4
  - search.c is at 5
- Get info on a particular file or directory

```
% svn info .
Path: .
Url: file:////tmp/repos/demo
Revision: 4
Node Kind: directory
Schedule: normal
Last Changed Author: blair
Last Changed Rev: 4
Last Changed Date: 2002-08-08 12:20:18 -0700 (Thu, 08
  Aug 2002)
```

How do Revisions work? (Part 5)

- Now Felix updates document.c
- Update working copy

```
% svn update
U  ./document.c
At revision 6.
```

Subversion Branch/Tags

- Uses “Cheap Copies” similar to Unix hard links.
  - Instead of making a complete copy in the repository, an internal link is created, pointing to a specific tree/revision.
  - As a result branches and tags are very quick to create, and take up almost no extra space in the repository
- No special commands for branching or tagging.

```
% svn checkout file:///c:/sun/myglassfish-svn ws
A ws/trunk...  (Checked out revision 340)
% cd ws; % svn copy trunk branches/my-branch
% svn status
A + branches/my-branch('+' means copy of something, not new)
% svn commit -m “creating private branch”
Adding branches/my-branch
Committed revision 341
```

Subversion Branch/Tags

- Creating Tag: Tag is just a "snapshot" of project in time

```
% svn copy file:///c:/sun/myglassfish-svn/trunk \
file:///c:/sun/myglassfish-svn/tags/release-1.0 \
-m "Tagging the 1.0 release of the project"

Committed revision 351
```

- Merging: "svn merge" is very close cousin of "svn diff"

  Instead of printing the differences between the revisions on terminal, it applies directly to local copy.

```
% svn merge -r 343:344 file:///a/b/trunk

 U integer.c

 % svn status

  M integer.c (copy of integer.c is patched)
```

svn switch

- Switching a working Copy:
  - “svn switch” - transforms an existing working copy into a different branch. Nice shortcut to users to change your working copy /a/b/trunk to mirror of new branch location.

```
% cd ws

% svn info |grep URL

URL: file:///a/b/trunk

% svn switch file:///a/b/branches/my-branch

U integer.c (Updated to revision 341)

% svn info |grep URL

URL: file:///a/b/branches/my-branch
```

- After “switching” to the branch, working copy is changed similar to fresh checkout copy of branch. Its more efficient to use this command as branches only differ by small degree.

Conflict Resolution

- `Look for the 'C' when you run svn update`
- Better than CVS
  - Conflict markers are placed into the file, to visibly demonstrate the overlapping areas. This matches CVS' behavior.
  - Three fulltext files starting with `tmp' are created; these files are the original three files that could not be merged together. This is better than CVS, because it allows users to directly examine all three files, and even use 3rd-party merge tools (as an alternative to conflict markers.)
  - Another improvement over CVS conflict handling: Subversion will not allow you to "accidentally" commit conflict markers, as so often happens in CVS. Unlike CVS, Subversion remembers that a file remains in conflict, and requires definite action from the user to undo this state before it will allow the item to be committed again.
- Solutions to resolve
  - Hand-merge the conflicted text
  - Copy one of the tmpfiles on top of your working file
  - Run `svn revert` to toss all of your changes
- Once resolved, you need to tell SVN that the conflict has been resolve
  - Run `svn resolved`
  - This deletes the tmp files

Log Messages (Part 1)

- Log messages are not embedded in files, like CVS
- Messages are associated with a single commit
- Possible to change log message after commit
- View log messages with `svn log` command

Log Messages (Part 2)

- See all log messages

  % svn log

  ---

  rev 3: fitz | Mon, 15 Jul 2002 18:03:46 -0500 | 1 line

  Added include lines and corrected # of cheese slices.

  ---

  rev 2: someguy | Mon, 15 Jul 2002 17:47:57 -0500 | 1 line

  Added main() methods.

  ---

  rev 1: fitz | Mon, 15 Jul 2002 17:40:08 -0500 | 2 lines

  Initial import

- Limit the range of log messages

```
% svn log -r 5:19
...  # shows logs 5 through 19 in chronological order
% svn log -r 19:5
...  # shows logs 5 through 19 in reverse order
% svn log -r 8
...
```

- See logs for a single file

```
% svn log foo.c
...  # shows log messages only for those revisions that foo.c changed
```

Subversion Local Repo for Practice

- Familiarize svn by creating local repository
  - svnadmin create c:/sun/myglassfish-svn
  - svn checkout file:///c:/sun/myglassfish-svn c:/sun/myworkspace (You can find .svn file in new workspace)

    Checked out revision 0.
  - svn add bootstrap (Added bootstrap module recursively)
  - svn update maven.xml (Update the file and checkin)
  - svn commit -m “Updating maven.xml file” maven.xml
  - svn copy maven.xml newmaven.xml
  - svn commit -m “Copying file” newmaven.xml
  - svn move newmaven.xml mymaven.xml
  - svn commit -m “Moving file” newmaven.xml mymaven.xml
  - svn log mymaven.xml

Subversion through Netbeans

- Subversion module is currently available for 5.5 and 6.0(dev) versions of NetBeans IDE.
- http://subversion.netbeans.org/faq/index.html#features
- Demo
- Download TortoiseSVN tool for windows which integrates svn into Windows Explorer.
  - http://tortoisesvn.net/downloads

References

- Subversion home

  http://subversion.tigris.org/

- Subversion quick reference guide/book

  http://svnbook.red-bean.com/

  http://subversion.tigris.org/files/documents/15/177/foo.ps

- Subversion source code and binary downloads

  http://subversion.tigris.org/project_packages.html

- CVS to SVN Crossover Guide

  http://svn.collab.net/repos/svn/trunk/doc/user/cvs-crossover-guide.html