

Using Modern Revision Control Systems, or “Saving All Your Work Forever”

Michael R. Head¹

¹Binghamton University

Binghamton University, Computer Science Graduate
Seminar Series

Outline

- 1 Motivation
- 2 History of the Tools
- 3 Current Generation Tools

Use-cases for revision control

The single user case

- Never manually copy your files again
- Full change history
- Your projects are inherently backed up
- Add-on tools like “viewvc”
- Trivial integration with Eclipse and other tools

The group case

- Integrate your changes with others more simply
- See others' history of changes
- Easy to switch from a personal project to a group project

A Story About Working with Others

Downloading tarballs

- Fine if ...
 - Making no changes
 - New releases infrequent
 - Tarballs are built properly
- Still want CVS for your own changes (libxml, piccolo, freeloader)
- CVS makes it relatively easy to support this with its `import` command
- Would still be much nicer have access to their repository

Storing Your Source Code... And It's All Source Code

Code

- C, C++, Java, C#, sh, ...
- \LaTeX (documents and presentations)

Other Text Files

- “Dotfiles”
- System config files

Binary Files

- Office/OpenOffice files
- PDFs from elsewhere

Short Demo

Eclipse

- 1 Share a project
- 2 Check out a project
- 3 Examine project history

ViewVC

- 1 Browse projects
- 2 Show a couple diffs

(Free) Historical Tools

RCS (Revision Control System)

- From the 80s
- Operates on individual files only
- Uses `diff` to store changes into individual files
- Still in use in some Wikis

CVS (Concurrent Versions System)

- Originally released in 1986 as a wrapper around RCS
- Has the concept of “branches”
- Has many wrapper tools (TortoiseCVS, ViewVC, CvsGui)
- Simple repository format – text (“,v”) files in directories
- Still ultimately works on individual files, ignores symlinks, and is ignorant of renames

The Second Generation (Subversion)

SVN (Subversion)

- Originated by some CVS maintainers
- First released in 2001
- Designed to overcome some basic limitations of CVS
 - “Atomic commits”
 - “Rename tracking”
 - “Directory versioning”
 - “Versioned metadata”
 - “Cheap operations are cheap”
 - Built-in web support
- Doesn't change mode of operation over CVS
 - Requires one central repository for all users to access

Department Resources

Available CS department repositories

- `:ext:alpha.cs.binghamton.edu:/opt/share/grid/cvs`
- `https://www.cs.binghamton.edu/svn/$USER`

Create your own

- `cvs init $HOME/cvs`
- `svnadmin create $HOME/svn`

Limitations of the Centralized Model

Technical

- Impossible to do disconnected commits (airplane mode)
- Requires server access even to start a new branch
- Requires a repository to start tracking changes
- Merging branches is a pain (esp. with CVS)

Administrative

- Requires a server (and admin) to host developer accounts

Social

- All the cool kids are using distributed tools

Working With Others

Getting Access

- Even if others were using CVS...
 - Need an account on the system
 - Must decide whether to deal with a branch or everyone commit to a HEAD
 - No control over remote system (backups, downtime)
- If a Bazaar branch were in use, I could just clone the branch and start working and publish my changes here

What a DRCS Provides

Major benefits

- Easy to “branch” a project
 - Pull in someone else’s project
 - Make changes, commit them, publish them to your site
 - Upstream easily integrates
- No server/admin support required to share your project

Additional benefits

- Disconnected commits trivial
- Most operations are faster (local repository)
- (Usually) trivial to publish a branch
 - Just requires sftp access to a web site

Torvalds's Offering

GIT

- Linux Kernel had been developed using Bitkeeper
- The Bitkeeper license was revoked, so Linus wrote a new tool
- Built to support the kernel team
 - Centered around patch incorporation
 - `git-bisect` very cool for tracking down bugs
 - Supposed to be very fast
- <http://www.kernel.org/pub/software/scm/git/>

Using GIT

```
$ echo hello > foo
$ git init
$ git add foo
$ git commit

$ git clone -bare . ../repo
# Requires git and shell access on subfire.org
$ scp -r ../repo subfire.org:www
$ ssh subfire.org chmod a+x www/repo/hooks/post-update

# push changes to the repository
$ git push subfire.org:www/repo
# pull changes from the repository
$ git pull http://subfire.org/~burner/repo
# check out the contents of the repository
$ git clone http://subfire.org/~burner/repo
```

Canonical's Offering

Bazaar

- Originally a fork of GNU Arch (a mostly abandoned early DRCS tool)
- Now written in Python
- Supports a large number of plugins
- Designed/evolved for ease of use
 - Supports centralized-style of work if desired
 - Generally only need a few commands: `branch`, `pull`, `push`, `commit`, and `merge`
- Free bazaar-based project hosting for any open source project at <http://code.launchpad.net>

Using Bazaar

```
$ echo hello > foo
$ bzr init
$ bzr add foo
$ bzr commit

# push changes to (or create a) web repository
# this could be bzr+ssh:// instead of sftp://
$ bzr push sftp://subfire.org/home/burner/www/repo
# get new updates from the repository
$ bzr pull http://subfire.org/~burner/repo
# create a new working directory from the repository
$ bzr branch http://subfire.org/~burner/repo
```

Bazaar also supports checkout, update, and commit, which work just like CVS with a remote repository

Other Offerings

Other free distributed revision control systems

- **Codeville:** <http://codeville.org/>
- **Darcs:** <http://darcs.net/>
- **Mercurial:** <http://www.selenic.com/mercurial/>
- **Monotone:** <http://monotone.ca/>
- **SVK:** <http://svk.bestpractical.com/>

Arguments for Sticking with What You Know (CVS)

“Good” arguments

- What you’ve got works
- Some third party tools are well tested and stable for CVS (TortoiseCVS, ViewVC, commit emailing, Eclipse)
- “I’m working with a team that only uses CVS”

Weaker arguments

- “I’ll never share my project out”
- Instability of core tools
- “I’ve got too much code in CVS”
- “Tools are too complicated”

Summary

- Storing the history of your work is a good thing
- Older generation tools, such as CVS and SVN are perfectly good for many use-cases
- Distributed revision control tools do offer significant advantages