The Mercurial SCM

Fast.
Simple.
Distributed.

Bryan O'Sullivan
bos@serpentine.com

Revision control: a crowded field

Accurev

BitKeeper

CCC/Harvest

ClearCase

Perforce

StarTeam

Surround

Vault

...

Aegis

Arch

Bazaar-NG

CVS

Darcs

git

PRCS

Subversion

SVK

Vesta

...

One way to look at things

- I have work to do!
- My SCM tool should:
  1. Be easy to understand
  2. Help me to work with others
  3. Let me work efficiently
- I want something *simple* that *works*

One way to look at things

- My SCM tool should:
  1. ***Be easy to understand***
  2. Help me to work with others
  3. Let me work efficiently

Be easy to understand

- **User quote:** “Mercurial's conceptual model is clean and simple enough to carry around in my head”

- Let's introduce three concepts:
  - Repository
  - Changeset
  - Working directory

What's a repository?

- Simple
  - A directory containing the history of my project
  - No fancy database, no big server: just a directory
- Lightweight
  - Making a copy (a “clone”) of a repository is cheap
- Everywhere
  - Each person works in their own repositories

What's *in* a repository?

Mercurial doesn't actually expose these details.

(But they're simple, and it helps to know what's going on.)

- Changelog
  - The history of changes to the repository
- Manifest
  - History of file versions used in each changeset
- Per-file data
  - History of every file that Mercurial tracks

Contrast the models

Traditional SCM

- Exactly one central repo
- Server acts as bottleneck
- Managing server load is expensive or impossible
- Distant users see slow response
- Server failure is catastrophic
- Net connection required

Mercurial

- Central repo is optional
- Servers used infrequently
- Put mirrors wherever you want, for free
- Distant users have everything at their fingertips
- Every repo is a full backup
- Fully productive anywhere

What's a changeset?

- A snapshot of the project at a point in time
- It records:
  - Who made the change (the “committer”)
  - A readable description of the change
  - What files were changed, what the changes were
  - What the parent changeset was
- Creating a changeset is called “committing”i t

What's the working directory?

- A view of the repo as of some changeset
  - This changeset is the working directory's *parent*
- I can edit any file in the working directory
  - My changes will get rolled into the next changeset
  - I can add, remove, rename, and copy files
- I can see what I've changed, and how

Micro-tutorial: Hg in 60 seconds

- Create a repository — `hg init myrepo`
- Go in there — `cd myrepo`
- Edit a file — `emacs myfile`
- Tell hg to track the file — `hg add myfile`
- Now what's happening? — `hg status`
  - "File has been added" — *`A myfile`*
- Record my changes — `hg commit`

One way to look at things

- My SCM tool should:
  1. Be easy to understand
  2. ***Help me to work with others***
  3. Let me work efficiently

Working with others

- People naturally work in parallel
  - Most revision control tools make this *hard*
- I make some changes
- I go to check them in ("commit" them)
- What if someone else committed first?

A common problem

- What if someone else committed before me?
- I have to merge their changes *before* I can commit
- **There's no permanent record of my changes yet!**
- A mistake during the merge can **lose my work**

The Hg model: branching

- Remember that a changeset has a parent?
- Two changesets with the same parent make a *branch*
- That's *all* a branch is!
  - Nothing dramatic or complex

The Hg model: merging

- What do we do with branches?
- Some changesets have *two* parents
- These are *merge* changesets
- A merge changeset simply says:
  - "This is how I merged changesets A and B"

Help me to work with others

- Mercurial *naturally supports* parallel work
- I commit my changes when it suits me
- My changes are clean and self-contained
- I don't merge with your changes until *after* I've committed mine

Merging without stress

- What if I make a mistake during a merge?
- My changes are still there; so are yours
- **No work gets lost**
- I simply redo the merge

Mercurial makes sharing easy

- Built-in web server
  - CGI server for Apache integration
- Use ssh (“secure shell”) for secure remote access
- Works over network filesystems
- Share work offline using email

Sharing is symmetric

- I **clone** a remote repo to get a local copy
- I **pull** new changes from a remote repo
- I **push** my changes to another repo
- After a push, the remote repo is identical to mine

Sharing mini-tutorial

hg commit *I'm finished with my changes*

hg push *Push my changes to another repo*

abort: unsynced remote changes! *Hg tells me I need to merge first*

hg pull *Pull remote changes into my repo*

added 1 changeset with 1 change to 1 file

(run 'hg heads' to see heads, 'hg merge' to merge)

hg merge *Merge their changes with mine*

hg commit *Commit the result of the merge*

hg push *Push my changes and the merge*

added 2 changesets with 2 changes to 2 files

One way to look at things

- My SCM tool should:
  1. Be easy to understand
  2. Help me to work with others
  3. ***Let me work efficiently***

Mercurial is very, very fast

- File architecture eliminates seeks
- Common operations are cheap
  - No need to wait for the software
- All data is local
  - No need to wait for the network
- Metadata is shared between local repos

Mercurial is space efficient

- A Linux kernel source tree is 262 MB in size
  - Just the files from a tarball, no history of any kind
- 1 year of Linux kernel history in Mercurial
  - 19,500 files tracked
  - 27,500 changesets committed (1,840 merges)
  - ~1,000 contributors
- Complete history: 293 MB

Some performance numbers

- Numbers from my 3-year-old laptop
- Used Linux kernel repo

Task	Time (secs)
Find changes in working directory	2.2
Commit change to one file	1.6
Annotate 3200-line file with 500-rev history	3.5
Make local clone	2.3
Populate working directory after clone	70.0
Pull 350 changesets over http	30.0

Mercurial makes me more efficient

- Simple concepts let me focus
  - Think less about SCM, more about work
- Commits and merges are separate
  - Harder to lose or corrupt work by accident
- Cheap repos let me sandbox my work
  - One repo per task
- Local repos let me work anywhere
  - On train, over slow net connection, you name it

Other useful things to know about

- Web support includes history browser
- Comprehensive hook support
  - Automate important actions (builds, code checks)
- Add optional features using extension modules
  - GUI tools
  - Patch queue management
  - Automated regression search

It can't *all* be roses. Right?

- Two major missing items (both soon to come):
  - Support for merging changes across renames
  - Comprehensive user manual
- Want to add:
  - Better GUI interfaces on Windows, Linux, MacOS
  - Integration with popular IDEs

Quotes from April user survey

- “The developers are super-helpful.”
- “The community is great around Mercurial.”
- “The Mercurial community is more polite and helpful than most.”

More April user survey quotes

- “I consider Mercurial the best version control system on earth.”
- “I was up and ready to go with Mercurial in less than 5 minutes.”
- “Mercurial was extraordinarily easy to learn.”
- “I used to like CVS a lot. I can't imagine going back. Really.”