Subversioned System Configuration

holger.krekel@merlinux.de

at EuroPython July 2006, CERN/Geneva

versioning system configs

- (linux) distributions need customization and configuration
- multiple people (sysadmins, developer) want to modify text configuration files
- some changes need reviews from others to prevent bad side effects
- without tool support this is all fragile and hard to track!

directly versioning?

- Using subversion, bzr, ... to directly version system configs induces problems and limitation:
  - user/root separation of permissions
  - system directories turn into Working Copies
  - Working copies need to be group-accessable/writable etc.
- contradicts the goal of “minimal intrusiveness”

indirectly versioning!

- Write a frontend that delegates versioning operations to an underlying versioning system
- map “to-be-versioned” files into user-specific working copies
- bidirectionally transform ownership/permission info
- use features of underlying versioning system (notification, history, diffs)

vadm: using it

- perequesites:
  - svn installation (including svnadmin)
  - sudo rights for executing user
  - a repository (can be a fresh one)

vadm init file:///sysrepo/mysystem

vadm add/remove/commit/diff/log /path/to/configfile

simply use the subversion commands you know!

notifications

- Install a post-commit hook into subversion repo to signal admins/developers about system changes
- a daily cron job may commit any pending changes
- a file containing paths to be versioned

experiences

- We currently use vadm to control some 20 systems
- with daily auto-committing it has proven to be a valuable tool, lowering the barrier to make changes to a running system
- you can find out who edited particular lines
- we'd like to version on a "cluster" basis

Versioning clusters

- Versioning multiple “similar” machines:
  - distinguishing “per-system” and “cluster” changes
  - allow for multiple clusters in hierarchical order?
  - per-system would take preference
- a “cluster” vadm should only require ssh + local svn config

more cluster considerations

- remotely access system states/configs (without manually logging in)
- ensure low-latency approaches: the slower the tool gets the less it will be used
- don't require a server-side daemon other than sshd?
- but allow for centralized repositories

Suggested semantics

- a URL defines a group of versioned files
- each system has a stack of such URLs
- URLs can be marked:
  - **auto-update**: changes automatically copied to system
  - **manual-update**: changes need manual trigger

Other considerations

- versioning package installation information
- auto-versioning directories (signalling additions/removals)
- auto-commits: determining who likely made the change? Integration into Nagios to signal a pending commit/update
- Speeding up versioning of hundreds of system files
- Plugging in other versioning systems?

Development history

- in 2002/2003 initial steps by Jens-Uwe Mager and Holger Krekel
- rewrites and refinements until today
- source living at http://codespeak.net/svn/vadm/dist, GPL
- unit-tested on various levels (py.test)
- driven by demands from codespeak and other system administration

vadm future

- development happens on a “demand” basis
  - from ourselves/involved parties
  - from contributors/users
  - from external paying parties
- release planned in 2006, GPL license
- sysadmin training/support possible
- Contact at holger@merlinux.de