Use of Cfengine for Automated, Multi-Platform Software and Patch Distribution

David Ressman
John Valdés

Department of Astronomy & Astrophysics
The University of Chicago

Background

Our department had 100+ UNIX systems running over half a dozen flavors of UNIX with only 2 sysadmins manually managing systems. This led to 2 major problems:

- Old and various copies of software
- Unpatched operating systems

Needed some way to automate software and patch administration.

Requirements

- Software distribution and management
- Patch distribution
- Centrally controlled configuration
- Ease of use
- Low cost
- Security

Requirements (cont.)

- Flexible host configuration:

  customize software installation and configuration based on group, host or other installed software
- Pull software and patches
- Portable:

  SunOS, Linux, IRIX, etc.
- Autonomous

Design Overview

- Automated system to be controlled via a centralized configuration manager
- Software to be packaged, distributed via anonymous ftp, and installed locally with scripts
- Patches to be distributed via anonymous ftp and installed with scripts
- Software, patch and host information to be stored in a SQL database

Software Components

- Cfengine for centralized system configuration
- RPM for package management
- MySQL for storing software, patch and host information
- Perl for scripts (cfengine modules)

Cfengine

- Manages files and processes through actions:
  - copy
  - editfiles
  - processes
  - shellcommands
  - and so on...
- Uses classes (groups) to control actions
- Easily extended through modules

Cfengine Example

```
groups:
  tftpserver = ( bootserver )

editfiles:
  tftpserver::
        { /etc/inetd.conf
        UnCommentLinesContaining "tftp"
        DefineClasses "inetd"
        }

  !tftpserver::
        { /etc/inetd.conf
        HashCommentLinesContaining "tftp"
        DefineClasses "inetd"
        }

processes:
  inetd::
        "inetd"
        action=signal
        signal=hup
        inform=off
```

Using Cfengine

- Central server running cfd and holding master copies of cfengine config files.
- Clients contact server once an hour to check for updated cfengine config files (using cfengine's `copy` action).
- Clients then run cfengine using the config files.

Red Hat Package Manager (RPM)

- Ported to all systems of interest; single “spec” file can be used to build on all OSes
- Integrated FTP client
- Desirable package management features:
  - Simple command to install/remove/verify software
  - Tracks installed files
  - spec file “documents” source of software, how it was compiled, how it was installed, etc.
- Active development community

Using RPM

- Create spec file for each software package and build binary rpms for each OS and architecture
- Upload rpms to central ftp server and add to database
- Clients fetch and install rpms via custom cfengine software module

Handling Patches

- Every OS has a different patch packaging format(s); no way to standardize on a common format
- Use scripts to regularly download patches from vendors to central ftp server
- Add patch info to database
- Clients fetch and install patches via custom cfengine patch module

MySQL Host and Software Tables

Package Name	Default Version	Solaris Version	Linux Version	Solaris Build	Linux Build
acroread	4.05-1	NULL	NULL	1	1
tcl	8.1.1-1	NULL	NULL	1	NULL
pam_opie	1.1-1	1.0-2	0.21-3	1	1

Hostname	Package Name	Package Version
mypc	acroread	current
mypc	pam_opie	current
mysun	tcl	7.6-1

- Software table contains list of all software to be distributed.
- Host software table contains list of all software which should be installed on all machines.

Patch Table

Patch No.	OS	OS Version
107451-04	solaris	5.7
108528-02	solaris	5.8
lpr-0.50-5	linux	6.2

- Patch table contains list of all patches to be distributed.

Software Module

- Query MySQL database for list of software packages and versions to be installed.
- Generate list of software packages and versions currently installed ("rpm -q -a").
- Compare the two lists and install, delete and/or upgrade any packages as needed.
- Define a cfengine class for each software package that remains installed on the system.

Cfengine Software Class Example

```
editfiles:
  imapd::
        { /etc/inetd.conf
        AppendIfNoSuchLine "imap  stream   tcp      nowait   root
                /usr/sbin/tcpd   imapd"
        DefineClasses "inetd"
        }

  !imapd::
        { /etc/inetd.conf
        DeleteLinesStarting "imap"
        DefineClasses "inetd"
        }

processes:
  inetd::
        "inetd"
        action=signal
        signal=hup
```

Patch Module

- Query MySQL database for list of patches which should be applied.
- Compare list to list of patches currently installed.
- Download and install/update patches as needed.
- Define a cfengine class for each patch that gets installed.
- Track patches which fail to install.

Finishing Touches

- Currently implemented for Solaris and Linux (Red Hat); IRIX to be added soon.
- Integrated with Solaris JumpStart and Red Hat Kickstart for completely automated system installation, configuration and patching!
- Simple web interface for maintaining host and software database tables.

Summary

- Automating software and patch installation has greatly simplified our job.
- Using classes in cfengine with software and patch installation allows us to completely automate the install process
- Works great!

Availability

All our scripts are freely available at:

http://astro.uchicago.edu/~davidr/cfengine-tools/