Unix Infrastructure Management From Scratch

Mike Renfro

Tennessee Technological University

Tennessee Higher Education IT Symposium 2008

CATBERT: EVIL DIRECTOR OF HUMAN RESOURCES
I'D LIKE TO CHANGE MY JOB TITLE TO SOMETHING WITH "ARCHITECT" IN IT.
MY DREAM IS TO DO LESS WORK WHILE ALLEGEDLY BEING MORE VALUABLE.
THE BEST I CAN DO IS "CODE MONKEY."
HOW ABOUT "SOFTWARE SIMIAN"?
www.dilbert.com
scottadams@aol.com
3-9-08 © 2008 Scott Adams, Inc./Dist. by UFS, Inc.

© Scott Adams, Inc./Dist. by UFS, Inc.

Three Great Virtues of a Programmer (or Sysadmin) — Larry Wall

- Laziness
- Impatience
- Hubris

Laziness "The quality that makes you go to great effort to reduce overall energy expenditure. It makes you write labor-saving programs that other people will find useful, and document what you wrote so you don't have to answer so many questions about it."

I'm a part-time systems administrator. I don't have time to sit at individual systems and answer the same installation questions over and over. I don't have time to babysit installing applications on a few dozen systems, either.

Impatience “The anger you feel when the computer is being lazy. This makes you write programs that don't just react to your needs, but actually anticipate them. Or at least pretend to.”

Of course all you systems should stay current on security updates. Yes, you, too. What makes you think you're different from the other systems in this cluster? And don't email me 20 log anomalies I don't care about; just the ones I do care about.

Hubris “Excessive pride, the sort of thing Zeus zaps you for. Also the quality that makes you write (and maintain) programs that other people won't want to say bad things about.”

I'll be honest. I want to look better than the average administrator, and better than I've looked in the past. If one of my systems drops a hard drive, I want to replace it, reboot the system from the network, and maybe stick around a bit longer to configure it for automated updates (but if I can go home and configure that remotely, even better). And when it's done updating, I want my users to be unable to tell the system's been reinstalled.

The groundwork for bare-bones installations

- Version control
- The gold server
- Host install tools
- Ad hoc change tools

Ensuring a consistent domain

- Directory servers
- Time synchronization
- Authentication servers
- Network file servers
- File replication servers

After the initial installation

- Client file access
- Client OS and security updates
- Client configuration management
- Client application management
- Email
- Printing
- Monitoring

Version Control

- You need this. You'll be writing a lot of code that will describe how you want your systems configured. You'll be keeping a central repository of configuration files, configuration templates, and other important data. You want to be able to track changes when something breaks down the road.
- Unless there's already a compatible repository in-house, the simplest route is probably to set up a simple Subversion repository on the gold server's local filesystem, and back it up regularly.
- Wikipedia has a comparison of revision control software.

The Gold Server

- Serves files out via NFS, http, rsync, or whatever protocol your configuration management software uses.
- Passive. Clients contact the gold server to pull down updates, rather than it pushing updates out to the clients.
- Not mission-critical. If it dies, configuration changes don't happen, but all clients continue to run in their current state.
- Often the OS installation server, the patch server, and the sysadmin server, but not always.

Host Install Tools

- Unattended installation of a vanilla operating environment is the goal
- May or may not want to include security updates at this point
- Add dependencies to allow the client to contact the gold server on reboot
- Probably want to use vendor-supplied installation tools (e.g. Debian preseeding, Redhat kickstart, Solaris Jumpstart)
- Probably don't want to use a system image (especially if you have a several hardware models or architectures)

Ad hoc Change Tools

- Traditional tools were rsh, rcp, and rdist
- Current preferences often include ssh or dsh
- If we do everything else right, we never use these for administration: only for troubleshooting, diagnostics, and forcing an immediate update from the gold server.

Directory Servers

- Maps hostnames to IP addresses, usernames to UIDs, group names to GIDs
- May be your responsibility to maintain, or you may leech off a larger organization
- Think long-term. Get CNAME entries in DNS for infrastructure roles: “cvs” for the version control server, “gold” for the gold server, etc., even if they point to the same physical systems at first. Then you don’t have to edit your scripts and definitions if you expand or rearrange your servers.

Time Synchronization

If you don’t synchronize all your system clocks:

- “make” acts funny when run over NFS
- “tar” complains about timestamps from the future
- You may have race conditions or other weirdness if you need multiple systems to coordinate on a particular task
- Kerberos just won’t work

Time Synchronization

- Just use NTP.
- Use the campus NTP if possible.
- Avoid any sync method that jumps the clock on a running system, rather than slowly adjusting it to a proper value.

Authentication Servers

- Users like having a single username and password.
- If you can leech off an existing authentication database (LDAP, Kerberos, Active Directory, NIS, etc.), do it.

Network File Servers

- Consider offering multiple methods to access the same files: NFS for Unix systems on a protected network segment, SMB or Appletalk elsewhere.
- Don't forget backup and restoration, too.

File Replication Servers

- Propagates master configuration files (or any other files) to clients.
- Caching filesystems are one option for homogenous infrastructures.
- Checkouts from a version control system are another option.
- Update and syncing programs like sup or rsync are another.
- You may need a way to automatically restart services after their config files change.

Client File Access

- Users and applications like a consistent namespace for files.
- If you can't be 100% consistent, at least be consistent within a particular system type.
- May use /apps as the main tree for software, full of links to real programs stored in /local/apps or /remote/apps as needed.
- Don't forget about access to users' home directories, too.

Client OS and Security Updates

This step desperately needs to be both automated and capable of being run unattended. Options include:

- Makefile that runs patch scripts in a well-defined order, and touches files to prevent patches from being run more than once.
- Vendor-supplied tools
- Free tools that use lower-level vendor-supplied tools

Client Configuration Management

- Everything that makes a client unique.
- Also, everything that a client has in common with other clients in a group (web servers) or in a common administrative domain.
- Includes configuration files, which services should be running, local users, locally-installed software.
- Can use a homegrown system, but it's probably worth using an existing tool.

Client Application Management

- End-user applications, usually third-party.
- Use native packaging formats (.deb, .rpm, .pkg) when possible.
- If multiple versions of a package are needed, you can install each to /opt/package/version and use symlinks in the default path to set a default version.

Email

- Just use SMTP.
- Open as few ports as possible—just because a system needs to email cron output doesn't mean it has to listen for outside emails.

Printing

- CUPS is a flexible printing system, compatible with JetDirect, Internet Printing Protocol, LPD, and Windows shares.
- Consider offering every printer available, and then setting a default printer based off the user's location.

Monitoring

- Central syslog server
- logcheck to notify about log anomalies
- Nagios for notifications of service or host problems
- Ganglia for trends of cluster performance and usage
- Cacti for trends of anything you can get into SNMP

The Goals

- Consistent user experience across all systems of a particular type (e.g., Solaris workstation, Debian cluster node, Ubuntu workstation) — bonus if we can make things consistent across system types, as well
- Authentication and authorization using existing Active Directory
- Restricting access to selected users in the Active Directory (primarily faculty, staff, graduate students)
- Less-stressed sysadmins

Version Control

- We had an existing Apache Subversion server, so we just made a repository for the gold server's data.
- Excerpts from Apache's httpd.conf and AuthzSVNAccessFile are attached at the end of the handouts.

The Gold Server

- Virtual server with 128 MB RAM, 10 GB disk
- Four open ports: ssh, Nagios plugin, rsync, and puppetmaster
- Puppetmaster's configuration files are a Subversion working copy

Host Install Tools (Debian)

- Existing Debian DHCP server extended to allow PXE booting
- All current PCs supported PXE
- Debian preseed configurations take up 2-4 KB per system type, and can be shared across architectures. In our case, three configurations describe nearly 100 nodes (everything except the main file server and ftp server).
- Debian systems PXE boot, grab configurations from FTP server, install minimal operating system and Puppet client from FTP server, and reboot. Under 10 minutes from initial powerup to login prompt on a Dell Opteron server, including formatting 650 GB of disk.

Host Install Tools (Solaris)

- Existing Debian DHCP/PXE server extended with Solaris configuration options
- Upgraded firmware on older Solaris workstations to allow PXE booting (good riddance to rarpd, bootparamd, and friends)
- Solaris Jumpstart configurations are a few KB each. Only one architecture in place for Solaris, so changes required for multiple architectures are untested.
- Solaris systems PXE boot; grab configurations from NFS server; install operating system, security updates, and Puppet client from NFS server; and reboot. Takes a few hours, due to larger operating system selection and inefficiencies in pkgadd.

Ad hoc Change Tools

- All clients had secure shell servers installed on them as part of the host installation stage.
- The gold server also has dsh (Dancer's shell / distributed shell) installed, which basically loops ssh runs over multiple systems in defined groups.
- The gold server keeps a master repository of client ssh keys. As soon as a client checks in via puppet's internal certificate authority, the puppet client can pull its key down and restart the ssh server. Users don't notice when a system has been reinstalled, since its key (effectively) never changed.
- The clients also install root@gold's public key into their root authorized key file, allowing dsh or ssh to connect without password prompts.

Directory Servers

For DNS, we use the campus DNS server.

nsswitch C library that allows username, password, and other lookups to query local files, network databases, etc.

pam Pluggable Authentication Modules for Linux, Solaris, and others. Authorization libraries for flat files, remote systems, etc.

winbind Samba component that generates Unix UIDs from Active Directory RIDs

Avoid local storage of account information for regular users.

```
ch208r:~# getent passwd mwr -s files
ch208r:~# getent passwd mwr -s winbind
mwr:*:6723:5513:mwr:/home/CAE/mwr:/usr/bin/scponly
```

/etc/pam.d/common-account (for Debian systems)

```
account sufficient      pam_unix.so
account required        pam_listfile.so onerr=fail \
  sense=allow file=/etc/security/users.conf item=user
account required        pam_winbind.so
```

Time Synchronization

- Using campus NTP server.
- NTP configuration controlled by puppet configuration management software. Puppet class for managing NTP is attached at the end of the handouts.

Authentication Servers

pam_krb5 PAM module that looks up passwords via Kerberos. Compatible with Active Directory.

```
auth     sufficient      pam_unix.so nullok_secure
auth     required        pam_krb5.so use_first_pass
```

Network File Servers

- Dell PowerEdge 2950, PowerVault MD1000, PowerVault 124T-LTO3, Debian GNU/Linux 4.0
- NFS access provided by regular nfsd, SMB access provided by Samba, Appletalk access provided by netatalk
- Amanda for backups and restores

File Replication Servers

This role has generally been replaced by application packaging and configuration management tools.

Client File Access

- Disk space has grown at a much faster rate than software bloat (at least in my case).
- Therefore, most packages are installed locally. Application packaging ensures identical paths.
- Configuration management tools take care of deciding which packages to install, and how to mount central file server stores.

Client OS and Security Updates

- For Debian, cron-apt runs “apt-get update” and “apt-get upgrade” to grab new security and major release updates. No new packages get installed, only new versions of existing packages. This also gets new versions of end-user applications.
- For Solaris, “pca” (Patch Check Advanced) runs nightly, grabbing new patches from Sunsolve.

Client Configuration Management

Puppet—`http://puppet.reductivelabs.com/`

- Open source system for automating system administration tasks. Written in Ruby, but Ruby knowledge not required for use. Most extensive support for Linux, Solaris, OS X, and BSD; lesser support for AIX, HP-UX, and other Unixes.
- Lead developer (Luke Kanies) in Nashville, available for training, support, and services.
- Puppet handles everything in "configuration management": distributing the right configuration files; starting, stopping, reloading, enabling, and disabling services; managing local users; managing packaged software; managing cron jobs; etc.
- The Puppet class for managing NTP at the end of the handouts is a decent introduction to the basics.

Client Application Management (Debian)

- Converted large binary-only software (ANSYS, MATLAB, Abaqus, etc.) to .deb format in a local package repository.
- To protect non-free packages from unauthorized access, root on clients connects to the repository as an unprivileged user via ssh, and copies packages down as needed. No http or ftp access required. ssh keys for this are distributed by puppet.
- Puppet natively supports “apt-get” for Debian packages.

Client Application Management (Solaris)

- Using Blastwave sites for free software.
- Local Blastwave repository for non-free packages to be added in the future.
- Puppet natively supports “pkg-get” for Blastwave packages.

Email

- ssmtp installed on most Debian systems.
- sendmail only listening on loopback interface on all Solaris systems.

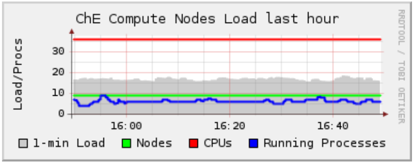
Printing

To be done later. So far, the major infrastructure management is still on systems that don't need to print.

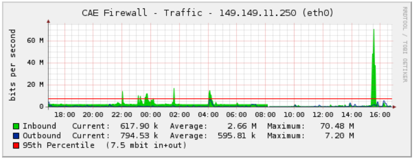
Monitoring

Monitoring is running, to various degrees of integration. Syslog, logcheck, and Ganglia are more integrated than Nagios and Cacti. Puppet manifests do exist that automatically add hosts and services to Nagios as they're defined, though.

Ganglia



Cacti



For More Information

- Mike Renfro, renfro@tntech.edu
- `http://infrastructures.org/`
- `http://puppet.reductivelabs.com/`
- `http://blogs.cae.tntech.edu/mwr/infrastructure-management/`

Apache Configuration for Subversion

Listing 1: Excerpt from httpd.conf

```
<Location /svn>
  DAV svn
  SVNParentPath /usr/local/svn
  AuthzSVNAccessFile /etc/apache2/AuthzSVN
  Satisfy Any
  Require valid-user
  AuthLDAPEnabled on
  AuthLDAPAuthoritative on
  AuthType Basic
  AuthName "CAE Subversion Repository"
  AuthLDAPBindDN cn=netauth,cn=Users,dc=cae,dc=tntech,dc=edu
  AuthLDAPBindPassword REDACTED
  AuthLDAPURL ldap://BDC:389/cn=Users,dc=cae,dc=tntech,dc=edu?\
      sAMAccountName?sub?
</Location>
```

Apache Configuration for Subversion

Listing 2: Excerpt from AuthzSVN

```
[groups]
cae-admins = mwr, jls
[goldserver:/]
@cae-admins = rw
```

Bare Minimum Puppet Example

```
# /etc/puppet/manifests/site.pp
import "classes/*"
node default {
    include sudo
}

# /etc/puppet/manifests/classes/sudo.pp
class sudo {
    file { "/etc/sudoers":
        owner => "root",
        group => "root",
        mode  => 440,
    }
}
```

NTP Class for Puppet (packaging, configuration)

```
class ntp {
  $ntppackage = $operatingsystem ? {
    Solaris => "SUNWntpu",
    default => "ntp"
  }
  package { $ntppackage:
    ensure => installed,
    provider => $operatingsystem ? {
      Solaris => "sun",
      default => "apt"
    }
  }
  file { ntpconf:
    path => $operatingsystem ? {
      Solaris => "/etc/inet/ntp.conf",
      default => "/etc/ntp.conf"
    },
    owner => root, group => root, mode => 644,
    source => "puppet:///files/apps/ntp/ntp.conf",
    require => Package[$ntppackage],
  }
```

NTP Class for Puppet (service)

```
  service { ntp:
    ensure => $virtual ? {
      vmware => stopped,
      xenu => stopped,
      default => running
    },
    enable => $virtual ? {
      vmware => false,
      xenu => false,
      default => true
    },
    subscribe => [Package[$ntppackage], File[ntpconf]]
  }
}
```

Puppet run example

```
ch208i:~# puppetd -vt --factsync --server gold.cae.tntech.edu
info: Loading fact virtual
notice: Starting Puppet client version 0.23.1
info: Retrieving facts
info: Loading fact virtual
info: Config is up to date
info: //ch208i.cae.tntech.edu/baseclass/ntp/File[ntpconf]: \
    Filebucketed to puppet with sum \
    afbf118cee5c7900d2d6e4856a30a581
notice: //ch208i.cae.tntech.edu/baseclass/ntp/File[ntpconf]/source\
    : replacing from source puppet://gold.cae.tntech.edu/files/\
    apps/ntp/ntp.conf with contents {md5}2\
    bdf28b5071d9403296e48d4aa5032ef
info: //ch208i.cae.tntech.edu/baseclass/ntp/File[ntpconf]: \
    Scheduling refresh of Service[ntp]
notice: //ch208i.cae.tntech.edu/baseclass/ntp/Service[ntp]: \
    Triggering ’refresh’ from 1 dependencies
notice: Finished configuration run in 7.96 seconds
```

Storing system-wide ssh keys

Storing these in a system-wide known_hosts file means an end to “The authenticity of host foo can’t be established. Are you sure you want to continue connecting (yes/no)?”

```
# /etc/puppet/manifests/classes/ssh-keys.pp
class ssh-keys {
    sshkey {
        "ch208a.cae.tntech.edu":
            alias  => [ "cvs.cae.tntech.edu", "cvs", "ch208a", \
                 "149.149.254.1" ],
            type   => "ssh-rsa",
            key    => "AAAAB3Nz ... s8H8mBhn0=",
            ensure => present;
    }
}
```

Installing JRE on Debian

```
# /etc/puppet/modules/jre5/manifests/init.pp
class jre5 {
  package { "sun-java5-jre":
    require      => File["/var/cache/debconf/jre5.seeds"],
    responsefile => "/var/cache/debconf/jre5.seeds",
    ensure       => latest;
  }
  file { "/var/cache/debconf/jre5.seeds":
    source => "puppet:///jre5/jre5.seeds",
    ensure => present;
  }
}
```

```
# /etc/puppet/modules/files/jre5.seeds
sun-java5-bin   shared/accepted-sun-dlj-v1-1    boolean true
sun-java5-jre   shared/accepted-sun-dlj-v1-1    boolean true
sun-java5-jre   sun-java5-jre/jcepolicy note
sun-java5-jre   sun-java5-jre/stopthread        boolean true
sun-java5-bin   shared/error-sun-dlj-v1-1       error
sun-java5-jre   shared/error-sun-dlj-v1-1       error
sun-java5-bin   shared/present-sun-dlj-v1-1     note
sun-java5-jre   shared/present-sun-dlj-v1-1     note
```