Playing with puppets

instead of managing your systems

Thomas Bellman

bellman@nsc.liu.se

National Supercomputer Centre
Linköping, Sweden

HEPiX spring 2009

What is Puppet?

http://reductivelabs.com/trac/puppet

A declarative language for defining the configuration of (Unix-like) machines.

Used to make sure that:

- packages are/are not installed
- config files have the correct contents
- services are/are not running
- and so on...

Short resource example 1

```
file {
    "/etc/ntp.conf":
        source => "puppet:///ntp/ntp.conf-client";
    "/etc/ntp":
        ensure => directory,
        owner => "ntp", group => "ntp", mode => 0755;
    "/etc/rndc.key":
        ensure => link,
        target => "/var/named/chroot/etc/rndc.key";
}
```

Short resource example 2

```
package {
    "ntp":
        ensure => installed;
    "dhcp":
        ensure => absent;
}

service {
    "dhcpd":
        enable => false, ensure => stopped;
    "ntpd":
        enable => true, ensure => running;
}
```

All resource types

* augeas
* cron
* exec
* file
* filebucket
* group
* host
* k5login
* mailalias
* maillist
* mount
* notify
* package
* resources
* schedule
* selboolean
* selmodule
* service
* ssh_authorized_key
* sshkey
* tidy
* user
* yumrepo
* zone
* zfs
* zpool
* computer
* macauthorization
* mcx
* nagios_command
* nagios_contact
* nagios_contactgroup
* nagios_host
* nagios_hostdependency
* nagios_hostescalation
* nagios_hostextinfo
* nagios_hostgroup
* nagios_hostgroupescalation
* nagios_service
* nagios_servicedependency
* nagios_serviceescalation
* nagios_serviceextinfo
* nagios_servicegroup
* nagios_timeperiod

Grouping resources into classes

```
class timeclient
{
    package { "ntp": ensure => installed; }
    file {
        "/etc/ntp.conf":
            source => "puppet:///ntp/ntp.conf-client";
        "/etc/ntp":
            ensure => directory, owner => "ntp", group => "ntp",
            mode => 0755, require => Package["ntp"];
    }
    service {
        "ntpd":
            enable => true, ensure => running,
            subscribe => File["/etc/ntp.conf"];
    }
}
```

What to apply where

```
class timeserver { ... }
class timeclient { ... }
class dhcpserver { ... }

node armstrong
{
    include timeserver
    include dhcpserver
}

node goodman, miller, ellington, basie
{
    include timeclient
}
```

Running Puppet

- Standalone
  - Using local files (manifests)
    (where "local" can be e.g. NFS-mounted filesystem)
  - Typically run from cron
- Client-server
  - Client daemon connects to server (Puppet master) at regular intervals
  - Master compiles manifests for client
  - Protocol is XMLRPC over HTTPS
    (Next version will use REST over HTTPS)
  - Authentication of both client and server using X.509 certificates

Why use Puppet?

- Automation
- Documentation
- Version control of *all* configuration
- Share and reuse
- Off-line system administration
- Structure

Good points about Puppet

- Regular, and human-readable, syntax
- High-level resource specifications
  - define host entries instead of editing /etc/hosts
  - define services to run/not run instead of explicitly calling chkconfig/svcadm/...
- Automatically uses yum, apt, chkconfig, svcadm, ..., depending on operating system
- Dependencies between resources to get proper ordering
- Powerful templating system for file contents
- Parametrized macros
- Extendable (in Ruby)
  Write your own resource types, and other extensions
- Active user community
- Responsive developers

Limitations

- Documentation is spotty
  - Wiki
  - Lots of recipies and "howtos", but too little specifying all the nitty-gritty details
  - Some pages are outdated
- You still need to know the names of packages, services and so on on different operating systems.
- Fairly young tool
  - Still evolving
  - But fairly few regressions
- Low-level file editing is lacking
- Internal web-server of the Puppet-master daemon doesn't scale well
  - But can be solved by running it under Apache or Nginx

Our uses of Puppet so far

- Two compute clusters
- Half a storage cluster
  The new disk servers in it have been configured using Puppet, but the older servers haven't been converted.
- My laptop :-)
- Internal servers (email, DNS, web, ...) this summer

Things to consider when using Puppet

- Separate manifests for each cluster, or a single set to rule them all?
- What is configuration and what is data?
  - `httpd.conf` is configuration, but the published web pages are probably not.
  - `named.conf` is configuration, but what about the zone files?
  - Are user accounts configuration or data?
- Decide what to manage using Puppet.
  - Preferably everything, but some things may be difficult to do with Puppet.
  - Don't be afraid of writing custom extensions.
- Manage whole files, or only entries in them?

Case study: Joining two clusters

Existing systems:

- One new cluster configured entirely with Puppet
  - Dual disks in worker nodes with software RAID-0
  - Infiniband network
  - One "analysis node" with more memory and hardware RAID-0 on four external disks
  - CentOS 5
- One old cluster (not using Puppet)
  - Single disks in worker nodes
  - Gigabit ethernet
  - Two "analysis nodes" with lots of memory for running large jobs interactively
  - CentOS 4

Both located across town, with no internet access.
Local sysadmins perform daily administration.

Joining two clusters, continued

The mission:

- Upgrade the user software environment on the old cluster to become identical to the new cluster.

The method:

- Connect the ethernet switch in the old cluster to the switches in the new cluster
- Re-install the old worker nodes with CentOS 5 and configure with Puppet

Joining two clusters, continued

The joined cluster has many types of nodes:

- New worker nodes
- Old worker nodes
- New analysis node
- Interactive login nodes
- File servers
- Cluster server (Torque, DHCP, Puppet-master, ...)

Lots of configuration is common among node types.

But enough differences that having a single node image using e.g. System Imager would probably become bothersome.

Joining two clusters, continued

Success!

- The joining of the two clusters was very smooth.
- A single cluster with identical software environment on all nodes.
- Changes only have to be done once.
- Much of the configuration was done in advance, without access to the clusters.
- Only real problems were due to buggy BIOSes and buggy firmware in an ethernet switch.