Configuration Management Through Puppet

Robert Gallagher
HEAnet Conference 2014
Snowflakes

- A server which...
  - Is difficult to reproduce.
  - Performs a unique function.
- Everybody has at least one of these:
  - And you'll “only ever need one”, right?
- Mabye it was setup by someone who left years ago.
- **Mabye it just fell out of the sky.**
- Little to no documentation, or worse, incorrect documentation.
Snowflakes

- Snowflake servers lead to multiple issues:
  - Hard to configure HA/resiliency.
  - Hard to upgrade.
  - Hard to apply security patches.
  - Forklift upgrades..at best:
    - At worse..panic upgrades.
  - 09:53:54 up 662 days, 20:08, 0 users, load average: 6.91, 6.11, 4.6
In order to break away from snowflake servers, a different approach is required.

Multiple, identically configured hosts that deliver the same service (roles).
• Managing large infrastructures by hand clearly won't scale.

• A configuration management system lets us establish a single, authoritative source for system configurations:
  – Abstracted from the OS.
  – Changes can be deployed to 1000 machines just as easily as 1.
  – Policy compliance is enforced.
• Side benefits:
  – Integration with existing tools.
  – Infrastructure as code (more later).
  – Version control.
  – Audit trails.
  – Reporting.
Early attempts in HEAnet:

- A proto-config-management-system has existed for some years.
- Basically a collection of shell scripts, perl and cron jobs. Very task-specific and fragile.

We aimed to replace this.
Infrastructure as ...

- Wiki Documents?
- Word Documents?
- Rumour?
• Programmatical define your systems:
  – Also known as DevOps.
• Resource Abstracion Language (RAL):
  – Define particular system resources (eg: files, packages, users) independently of the underlying OS.
  – Describe the desired state of the system in your RAL.
  – It's up to the config management system to bring everything into line.
• CFEngine:
  – One of the oldest and most established offerings.
  – Introduced the concept of the RAL.
  – Fast and lightweight (written in C).
Chef:
- RAL largely based on Ruby.
- More Dev than Ops.
- Geared towards application deployments.
• Ansible:
  - Script-based approach with “Playbooks”.
  - Ansible runs are initiated from a central host.

• Salt:
  - Agent (minion) runs on each host.
  - Minions connect to a salt master, which can send out commands to be executed remotely or configuration directives to be applied.
• Why Puppet?
  – Quite a high level RAL in the form of *manifests*.
  – Very active community.
  – Ability to define *environments*.
  – Strong dependency management.
  – Large surrounding toolset.
Puppet Components

- **puppetmaster:**
  - Distributes manifests to agents on demand.
  - *Environments* give the ability to serve out different versions of code to your agents.

- **puppet agent:**
  - Implements manifests (locally, or from puppetmaster).

- **facter:**
  - Gathers metadata about the host, which can be used in manifests.
**Puppet Components**

- **Puppetdb:**
  - Collects data generated by puppet agents, for use in other puppet manifests.

- **Hiera:**
  - Define hierarchical data for use in puppet manifests.

- **Puppet ecosystem:**
  - **Foreman** – Stats, audits, server lifecycle control.
  - **MCollective** – Parallel job execution framework.
  - **Blueprint** – Generate manifests from existing system.
  - **Augeas** – Edit legacy config files.
group { 'heanet':
    ensure => present,
}

user { 'heanet':
    ensure => present,
    home => '/home/heanet',
    gid => 'heanet',
    comment => 'HEAnet role account',
    shell => '/bin/bash',
    password => '*',
    require => Group['heanet'],
}

file { '/home/heanet/
    ensure => directory,
    owner => heanet,
    group => heanet,
    mode => '0755',
    require => User['heanet'],
}
service { 'anycast-dns-monitor':
    ensure  => running,
    name    => anycast-dns-monitor,
    hasstatus => false,
    enable  => true,
    subscribe => File['anycast-dns-monitor.cfg'],
}

service { 'quagga':
    ensure  => running,
    name    => quagga,
    hasstatus => false,
    enable  => true,
    subscribe => [
        File['quagga-daemons'],
        File['quagga-bgpd.conf'],
        File['quagga-zebra.conf']
    ],
}

service { 'unbound':
    ensure  => running,
    name    => unbound,
    hasstatus => false,
    enable  => true,
    subscribe => File['unbound.conf'],
}
<%= scope.lookupvar('standard_vars::puppet_header') %>
!

! Zebra configuration for anycast DNS node <%= @hostname %>. !

hostname <%= @hostname %>
password cantcrackthis
enable password cantcrackthis
log file /var/log/quagga/zebra.log
!
<% interfaces_array.each do |interface| %>
interface <%= interface %>
  ipv6 nd suppress-ra
!
<% end %>
interface lo
!
!
line vty
!
file { 'quagga-zebra.conf':
  ensure   => file,
  path     => '/etc/quagga/zebra.conf',
  require  => Package['quagga'],
  content  => template('anycast_dns_node/quagga-zebra.conf.erb'),
  owner    => quagga,
  group    => quagga,
  mode     => '0640',
}
Puppet Modules

- Puppet manifests, files and templates can be packaged into a **module**.
- **Modules** are intended to be self-contained and reusable:
  - Ntp server module.
  - Apache server module.
- The puppet community publishes an extensive collection of modules on the **puppet forge**:
  - [http://forge.puppetlabs.com](http://forge.puppetlabs.com)
Demo

Let's All Go To The Lobby

To Get Ourselves A Treat!

https://github.com/spoofedpacket/conference2014
Lessons

• Your infrastructure is now code, so..
  – Put your manifests under version control.
  – Never allow changes to be made on the live puppetmaster.
  – Codify deployment procedures.
Lessons

• The puppetforge contains a large collection of modules.
  - Chances are, someone else has already done what you are trying to do.

• Separate data from code with hiera.
  - Hiera data can also be put into version control.

• Partition your environment into staging, production, etc.
Puppet in HEAnet

- Puppet in heavy use since early 2012, controlling around 206 hosts.
- Foreman for node classification.
- Module data stored in foreman, hiera and the modules themselves.
- Puppet code stored in git, 1,340 commits so far.
(Near) Future Plans

- Finally retire old shell-script based system.
- Use foreman to automate build processes.
- Mcollective for task automation.