Puppet at USCMS-T1 and FermiLab (and Beyond)

Tim Skirvin
USCMS-T1 @ FNAL
tskirvin@fnal.gov

Supported in part by the Department of Energy
DE-AC02-07CH11359

A Few Things About Puppet

- I'm not going to describe the tool in detail.
- It's not the only solution out there, but it's a pretty good one.
- Has the strengths and weaknesses of a programming language (but it isn't one)
- Puppet and Ruby are both moving targets
  - Example: 'include' has been deprecated and then un-deprecated over the last ~18 months
  - Collaboration with the rest of the world depends on following then-current shared best practices
- Puppet Labs is facing the startup "Market Share vs Income" inflection point
  - They're pushing Puppet Enterprise; we don't want it.

Another Thing About Puppet: Cattle vs Pets

Service Model

- Pets are given names like pussinboots.cern.ch
- They are unique, lovingly hand raised and cared for
- When they get ill, you nurse them back to health

- Cattle are given numbers like vm0042.cern.ch
- They are almost identical to other cattle
- When they get ill, you get another one

- Future application architectures should use Cattle but Pets with strong configuration management are viable and still needed

Gavin McCance, CERN 17

Metaphor popularized by Tim Bell @ CERN

We all have to support both Pets and Cattle

Configuration Management @ FNAL

- Lots of teams with separate infrastructures
- Shared infrastructure is fairly generic: DNS, Kerberos, syslog service, Scientific Linux
- As of mid-2013, ~ 5 teams had transitioned to Puppet for configuration management
  - Mix of Puppet 2 and Puppet 3
  - svn or git for version control back-ends
  - Extremely varied design patterns and styles
  - Minimal code-sharing or collaboration between teams
    - This isn't the ideal state!

Configuration Management @ USCMS-T1 (as of Jan 2013)

- Three major “classes” of hosts
  - Workers: ~800 hosts (cattle)
  - Storage: ~300 hosts (mostly cattle)
  - Servers: ~100 hosts (mostly pets)
- All hosts run SLF5
- ROCKS for system installation
  - Provides configuration management by reinstallation
- bcfg2 for ongoing configuration management
  - Manages ~10 files, mostly consistent across all hosts
  - CVS backend

Configuration Management @ USCMS-T1 (goal state)

- Little change in worker-storage-server mix
- Migrate all hosts to SLF6
- Cobbler for system installation
- Puppet for ongoing configuration management
  - Newest version (Puppet 3) (no, we probably won't stay on the bleeding edge)
    - Puppet Dashboard for viewing reports
    - puppetdb backend for automated reports/queries
    - Separated out Certificate Authority
    - Load-balanced front-ends
  - Manages ~250 resources on a worker node
  - git backend
- Build local RPMs, rather than /opt + rsync

Puppet @ USCMS-T1

Training the Team

- Configured an initial environment for testing and (limited) production use
- Hosted on-site Professional Training
  - Puppet Fundamentals for System Administrators https://puppetlabs.com/services/training/puppet-fundamentals
  - 2x 3-day sessions, ~25 people total from FNAL (including all sysadmins from USCMS-T1)
- Started by working on the Cattle
  - Workers are simple conceptually, but have to be fully automated – no manual steps allowed!
  - Pets can be ~90% automated, but are more specialized.

Defining Best Practices

- http://docs.puppetlabs.com/guides/style_guide.html
  - We have a local style guide as well
- https://github.com/rodjek/puppet-lint
  - This can be hooked into your editor or IDE
- git pre-commit and post-receive hooks to validate code and templates
  - Per-user local pre-commit hooks to check syntax quickly
- What doesn't go into puppet?
  - Scripts and code should go into separate RPMs
  - "Secret" data (e.g. passwords, keytabs) should not go into the main code repository
  - Disks and networks are configured via cobbler

Creating an FNAL Puppet Community

- CMS is extremely interested in sharing code and experience with our peers
- Created a central puppet-users@fnal.gov community and mailing list at Fermi
  - Mostly a fork from linux-users team run by Pat and Connie
  - Allows everyone to speak a common language
  - Gives us a place to share experiences and tools
    - …I guess I don't have to explain that to HEPiX!
  - Gave us the clout to bring in professional trainers
- Semi-Regular meetings (~1/month)
  - Lunch-and-learns; semi-formal meetings; whatever comes up

Puppet Environments + Git

- https://puppetlabs.com/blog/git-workflow-and-puppet-environments
- Individuals can create their own git branches for development and testing without intefering with the “production” environment
- Ready-to-test changes go into an “itb” branch, which is tracked by development hosts
- Tested changes can be promoted to “production” based on standard code promotion techniques.
- This has proven to be extremely useful!

Hiera

- http://projects.puppetlabs.com/projects/hiera
- Provides a mechanism for separation of code and configuration data within the Puppet tree.
- Can provide data out of a variety of back-ends: yaml, json, databases, .gpg files (for passwords)
- Makes it much easier to provide per-host or per-role defaults and overrides of standard behaviour
- This is fairly new, and not fully documented yet, but worth exploring.
- Frankly, this is just good. Use this if you can!

Puppet Roles + Profiles

- http://www.craigdunn.org/2012/05/239/
- Provides a clear abstraction layer to distinguish puppet modules and host configuration.
- Every host has a single defined role
  - e.g. ‘gridworker’, ‘puppetmaster’, ‘ganglia’
- Each role has 1+ profiles
  - e.g. ‘database’, ‘webserver’, ‘condor client’
- Each profile loads puppet modules
  - e.g. ‘mysql’, ‘apache’, ‘condor::client’

Puppet Forge

- http://forge.puppetlabs.com/
- Provides a central repository of “semi-blessed” puppet modules (think Perl’s CPAN)
  - Plus: somebody else has written the code
  - Minus: their requirements probably weren’t the same as ours
  - Plus: somebody else is often maintaining the code
  - Minus: it’s difficult to integrate necessary local changes with upstream code, so it may be easier to just fork
  - Minus: publicly-usable code generally requires loads of possible parameters -> unwieldy codes
- We are using Forge modules where reasonable, but we recognize their limitations.

Puppet Labs Standard Library

- http://forge.puppetlabs.com/puppetlabs/stdlib
- Provides a library of tools to enhance puppet
  - Work around scope limitations: ensure_resource, ensure_packages, getvar
  - Work with arrays: flatten, grep, is_array
  - Data type validation: validate_array, validate_hash
  - Helper plugins: facter_dot_d
- You should be using this.

Local Puppet Module Development

- condor – start from scratch, need to share with the world
  - Support code fragments in /etc/condor/config.d
  - When to restart/reconfig puppet based on config changes?
  - Good candidate for sharing on the Forge
- eos, dcache – need to codify existing practices
  - Good candidates for sharing within FNAL (for starters)
- cvmfs – basing on code from multiple sources
  - CERN: https://github.com/cernops/puppet-cvmfs
  - UK GridPP: https://github.com/HEP-Puppet/puppet-cvmfs
  - Tyler (FEF@FNAL) maintains a local module as well
- puppet itself
  - Existing modules on the Forge were a good start, but didn't exactly match our needs
- So many more!

Cobbler

- http://www.cobblerd.org/
- Disk layouts are defined within Cobbler
- Networks are configured at install-time
  - Don't want to let Puppet break anything that it can't fix
- Wrote wrapper scripts to integrate existing CMDB data (MACs + IP addresses)
- XMLRPC API allows puppet CA to auto-sign certs based on Cobbler's to-be-installed database
- Everything else is managed by a Puppet run on initial installation.

Next Steps

Upcoming Challenges @ USCMS-T1

- More and better internal documentation
  - …and maybe share it with our collaborators?
- Scaling to ~1500 nodes
- Retire the ROCKS servers entirely
- How to avoid becoming a “Ruby Shop”
- Tracking Ruby and Puppet
- Balancing Cobbler vs Puppet
- Changing the mindset to separate pets vs cattle
- Better use of/more effective test suites

Upcoming Goals - Local

- Start contributing modules to the Forge
- Take advantage of community “best practices”, e.g. Vagrant for unit testing
- Bring in trainers for Puppet Advanced training
- Increased collaboration with the larger Puppet community

Working with the HEPiX Community

- https://twiki.cern.ch/twiki/bin/view/HEPIX/ConfigManagement
  - Oh look, this working group already exists!
- How can we best share our configuration management experiences with our community?
- Do we have code that we can share?
  - What needs to be shared, anyway?
  - If we are going to share, are we going to have standards?
- What about support tools: git/svn, cobbler, etc?
- How do we integrate non-HEP groups?