Lessons Learned in the OSUOSL Puppet Migration

Lance Albertson
Director
lance@osuosl.org
@ramereth
Session Summary

- OSL team environment overview
- Brief OSL systems architecture history
- CFEngine environment & stats
- Initial Puppet environments
- Git repo all the things!
- Modules strategy
- Deployment/migration strategy
- Future Plans

Lessons Learned in the OSUOSL Puppet Migration
Lance Albertson | lance@osuosl.org | @ramereth
OSL Team Environment

- 2 FTE Sysadmins, 1 FTE developer
- 6-10 student sysadmins
- 4-6 student devs
- Student productivity / turnover
- Multiple on-going projects
- Incoming tickets keeps us busy
- Domain knowledge is mostly on me*

* I'm bad at documentation
OSL Team Environment

Left to Right: Rudy (basic), Daniel (irdan) and Sean (chekka)

Lessons Learned in the OSUOSL Puppet Migration
Lance Albertson | lance@osuosl.org | @ramereth
OSL Systems History

- Started out on Debian ('03-'05)
- Switched to Gentoo Hardened ('05-'12)
  - Gentoo devs on staff (mostly me)
  - Wanted the grsec/PaX features
- Started deploying CentOS 5 ('08+)
- All new deployments CentOS 6 ('12+)
  - EOL schedule worked best for us
- CFEngine 2 for config management
- Some mixed CFEngine/puppet env.
CFengine Environment

- Manages all package installs & upgrades
- 180 "Services"
- 174 Package classes
- 110 cf. files
- 19,200 lines of raw cf files
- 14,700 lines actual code
- 1440 lines in cf.classes alone
- 23,000 commits (8,800 are mine alone)
CFengine Environment

- "Do all the things in cfengine"
- A lot of hacked logic because its CF 2
- Not very dynamic for our needs
- No way to delegate access to projects
- Upgrade path to CF 3 would be a nightmare
- But excellent-ish support for Gentoo :-)

Lessons Learned in the OSUOSL Puppet Migration
Lance Albertson | lance@osuosl.org | @ramereth
Reason for choosing Puppet

- Liked its goals and approach overall
- Proximity to Puppet Labs (PDX)
- Lots of sharable modules and code
- Excellent community
- Lots of progress in its feature set
- Horrible Gentoo support :-(
  - But its improving a little!
Puppet Migration Strategy

- Avoid mixed cfengine / puppet environments at all cost
  - Either all cfengine or all puppet
- Convert all CentOS hosts first
  - Easy transition
- Rebuild, Retire, Rearchitect Gentoo hosts
  - Same basic architecture since 2005
The Beginnings

● Planning in early 2010
● Summer of 2010 initial implementation
● Student project
● Used code from example42
● Single git repo (#1) for everything
● Gentoo Puppet issues
  ○ No concept of use flags, keywords, etc
  ○ Package dependencies are hell
Git repo all the things! - 2010

- Fall 2010 - Git repo #2 created
- Split into repos based on projects
  - Try to solve the delegation problem
  - Manifests were done in an ugly way
  - Didn't use submodules but a simple script to keep things in sync - i.e. non-standard
- Repo is in production still today
  - Drupal Project has their own module repo
  - They use it to manage their services primarily
  - We run CFengine along-side puppet on Drupal Project machines
Git repo all the things! - 2011

- Summer 2011 Repo #3 created
- 2-3 students started it again
- Partnered with PDXCAT team
- Repo for every module, and submodule everything
- Nightmare management of the super-repo
- Very confusing to new students
- Deployed it on all the student workstations to test
  - Now its instance #2 that is running in production
Git repo all the things! - 2011

- Summer 2011
- Added basic puppet syntax checking commit hook
- Implemented puppet-sync
  - https://github.com/pdxcat/puppet-sync
  - A script to synchronize you manifests from a GIT repository to your Puppet master.
- This helped but the submodules produced so much rage face
Git repo all the things! - 2012

● Summer 2012
  ○ I take a crack at fixing the repo mess
● Repo #4 is created with a slightly saner approach
● Still using submodules, but much less
● Refactored a bunch of code
● Did take a look at mr for repo management
● Never went into production, kind of forgotten
Git repo all the things! - 2013

- Fall 2013 - "Lets get this right for real"
- Very simplified multi-repo layout (K.I.S.S.)
- Single repo
- Directory layout:
  - dist/ - internal modules
  - libs/modules - public or internal->public modules
  - site/ - site specific modules
    - site/os - OS specific module
    - site/role - Role specific module
- Submodules for libs/*
Lessons learned from the repos

- K.I.S.S. from the start
- Submodules produces a lot of rage face
- Pick something and stick with it the best you can
- Flexibility is nice, but don't need it in the beginning
- Try to use community modules when possible
Current status of migration

- Workstations on new "simple" repo
- Building base modules (80% finished)
  - Importing some from the other repos, refactoring, cleaning up
  - Using community modules (mysql, concat, etc)
- Importing OSL site specific magic
  - Converting CFengine-isms into Puppet
  - Try and undo really hacked up code and processes
- Testing and more testing
Testing Environment

- Vagrant and more vagrant
  - Build standard cfengine-ized basebox
  - Run new modules to see changes
- Future Plans
  - RSpec testing on modules (eventually)
  - Jenkins CI environment of some kind
- Projects' access
  - Give them an environment to test major changes
- Needs a lot of work
Current Migration Strategy

● Finish base puppet modules
  ○ Build other modules as needed

● Deploy application service management
  ○ Services not currently managed by CFengine
  ○ Will run in parallel with CFengine

● Convert all current CentOS 5/6 hosts one-by-one

● Gentoo -> CentOS Migration
  ○ Rebuilt as CentOS 6 under puppet
  ○ Retired, re-architected, etc
Future Plans

● Publish OSL modules

● Delegation to projects
  ○ Allow projects to check out their puppet config
  ○ Integrated testing, merging, etc
  ○ Less work on our staff long term

● Use other tools
  ○ Deploy Foreman (or something similar like PE)
  ○ Try out Puppet Enterprise
  ○ Look into PuppetDB, MCollective, Hiera, etc
Conclusion Takeaways

- Dedicate someone on the conversion
  - Maintain consistency, less context switching
- Use K.I.S.S. principle as much as possible
- Target specific hosts as examples for conversion
  - Do test deploys with vagrant
- Use "brick and mortar" philosophy
  - Try and keep site specific code out of the modules the best you can

Lessons Learned in the OSUOSL Puppet Migration
Lance Albertson | lance@osuosl.org | @ramereth
Questions?

Lance Albertson
lance@osuosl.org
@ramereth
http://osuosl.org
http://lancealbertson.com

Follow OSUOSL
@osuosl | fb.com(OSUOSL
G+ "Open Source Lab"

This work is licensed under a
Creative Commons Attribution-Share Alike 3.0 United States License.
Copyright 2013