Linux Clusters Institute: Configuration Management

Jonathon Anderson, Associate Director Research Computing Technology, University of Colorado Boulder
About me

• HPC systems since 2006
• Started using Puppet in 2009
  • Also use SaltStack for personal projects
• CU Boulder Research Computing since 2014
Goals

• Understand what configuration management is and why it is useful
• Know what tools exist (and how to choose?)
• Be equipped to convey the benefits of configuration management to peers and management
Out of scope

• Learning everything you need to know about a specific tool
  • Puppet will be used in examples; but the principles are broadly applicable

• Designing a specific or complete configuration management strategy for your site
What is “configuration management”?  

• Every system has a current state  
  • Files on the hard drive  
  • Running processes and services  

• That state has to come from somewhere  
  • Installation / provisioning procedure  
  • Manual “by hand” changes or scripts run  
  • “Golden master” images
Features of modern systems

• Idempotency
  • “Desired-state” configuration

• Revision control
  • “Infrastructure as code”

• Composable and flexible
Why bother?

- Automation
- Composition
- Confirmation
- Revision history
Benefits of configuration version control

- Built-in documentation (change logs, summaries, etc.)
- Peer review (issue tracking, merge requests, email alerts)
- Reverts

http://infrastructure-as-code.com
Benefits of configuration management summary

- Centralized catalog of all system configuration
- Automated enforcement of system state from an authoritative source
- Ensured consistency between systems
- Rapid system provisioning from easily-composed components
Modern configuration-management systems

- Puppet
- Chef
- CFEngine
- Salt
- Ansible
Getting started

• Pick a simple, common part of your configuration
  • ntp
  • resolv
  • nsswitch
  • sudoers
• Implement and test (start with “no-op”)
Directory structure

modules/
  ntp/
    manifests/
      init.pp
    files/
      ntp.conf
class ntp {
  package { 'ntp':
    ensure => installed,
  }

  file { '/etc/ntp.conf':
    source => 'puppet:///modules/ntp/ntp.conf',
    owner  => 'root',
    group  => 'root',
    mode   => '0644',
    require => Package['ntp'],
  }

  service { 'ntp':
    ensure => running,
    enable => true,
    require => File['/etc/ntp.conf'],
  }
}
# manifests/site.pp
	node 'node1' {
	  include ntp

}
Testing the prototype

# puppet apply --noop \
   --modules modules manifests/site.pp
Next steps

• Top-level node roles
• Add features you need now (don't try to do everything at once)
• Convince, teach, and assist your team
• Continue until you have no more questions about your environment
Advocating to colleagues

- Work is front-loaded, so early work seems much more costly
- System might undo work done by others
  - Add comments at the top of managed config files
- Offer to help colleagues port
- Work with at least one other person
- Be as transparent as possible
  - Commit emails
- Document how to port an existing host
Advocating to management

• Work more efficiently (get more done)
• Not an all-or-nothing proposition: start with a few systems and go slow
• Document and report success stories
  • Deployment speed improvements
  • Patch deployment improvements
  • Peer review anecdotes
  • Corrections made
Things to watch out for

• Also easy to make a mistake on several hosts at once
  • Test in isolation first, and with a no-op mode
• It's easy to get lazy and allow systems to fall out-of-sync
• It's easy to let perfectionism take over