

LCI workshop

on high performance clustered computing

# Linux Clusters Institute: Configuration Management

Zhongtao Zhang, System Administrator, Holland Computing Center,  
University of Nebraska-Lincoln

# About me

- PhD in Computational Chemistry
- Started work as System Administrator in HCC from 2015
- Started using Puppet in 2015

# 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
```

```
# modules/ntp/manifests/init.pp

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
```

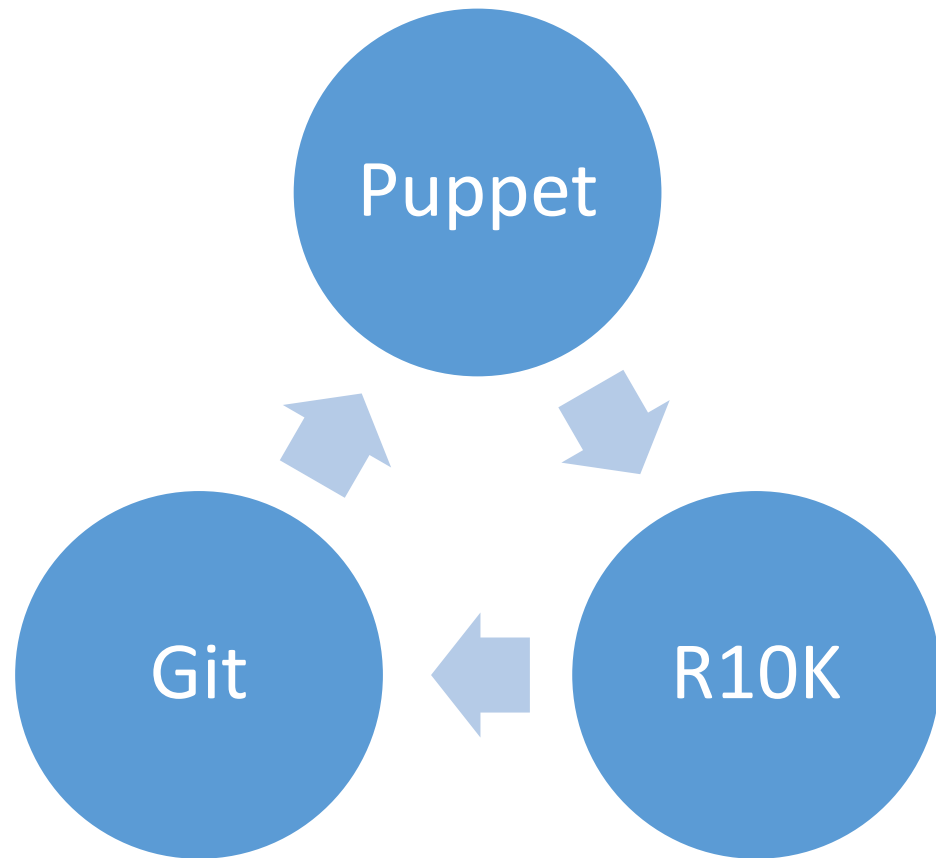
## Next steps

- 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
- Find more modules on <https://forge.puppet.com/>

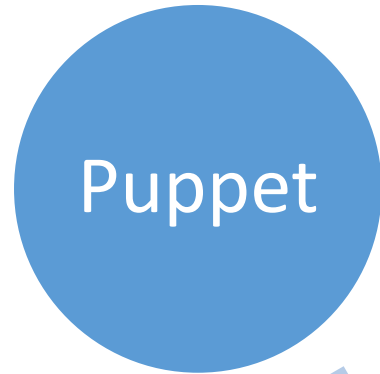


# Puppet workflow in HCC

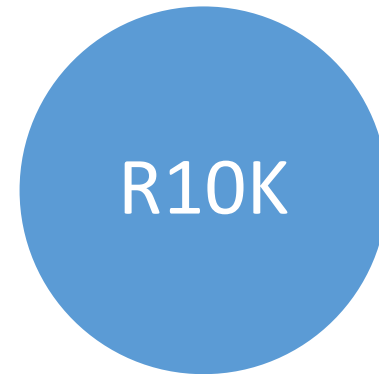
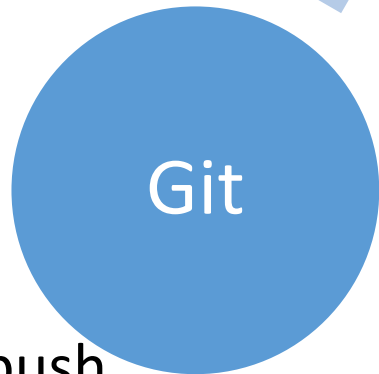
- Roles and profiles
- Hiera
- R10K
- Git



You can add more:  
Gerrit,  
Jenkins....



environment = production  
environment = test



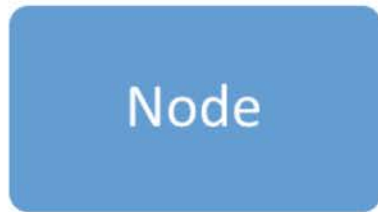
Puppetfile

clone, commit, push  
pull, merge....etc.

# What does this workflow look like?

- `git clone git@git-server:puppet`
- `git checkout -b mybranch`
- ... make some changes...
- `git add/commit/push`
- On you test node: `puppet agent -t --environment=mybranch`
- Merge it to production!

# Roles and Profiles



```
node head {
  include role::headnode
}
```

A node only have one role

```
class role {
  include profile::base }
class role::headnode inherits role {
  include profile::scheduler
  include profile:: webserver .....
```

A role defines a type of server: e.g. login node, head node, compute node' Can have multiple profiles

```
class profile::webserver {
  class { "httpd": }
  class { "php": }
  class { "memcache": }
}
```

A profile is a technology-specific wrapper.

# 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

## Reference

- Puppet: <https://puppet.com/>
- Puppet forge: <https://forge.puppet.com/>
- R10k: <https://github.com/puppetlabs/r10k>
- Roles and profiles: <http://garylarizza.com/blog/2014/02/17/puppet-workflow-part-2/>