InfoSeption

POLARIS live your dream

Puppet Workshop Training Notes (Installation and Configuration Guide)

Training dates	5th, 6th & 7th Sep2015
Place	Dee Cee Manor Hotel @ T Nagar
Trainer Name	Vishvendra Singh Chauhan from KOENIG EMAIL: vishvendra.singh01@gmail.com
Hosted by	Infoseption (PuppetLabs Authorised Solution Provider)

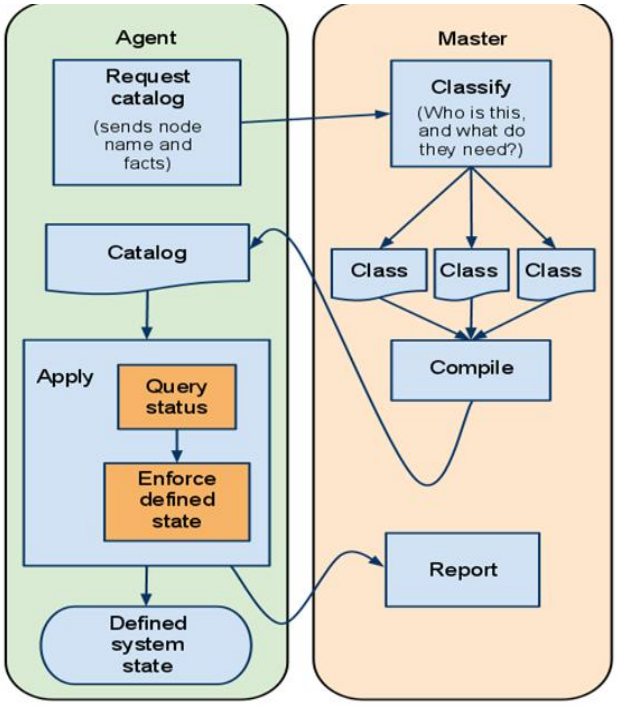
Puppet Overview:

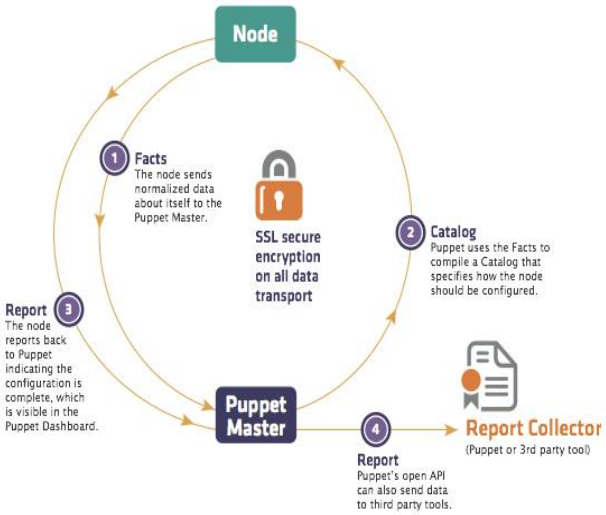
- Puppet is a configuration management system that allows you to define the state of your IT infrastructure, and then automatically enforces the correct state.
- Puppet automates tasks that system admins often do manually, freeing up time and mental space so system admins can work on the projects that deliver greater business value.
- Puppet automates every step of the software delivery process: from provisioning of physical and virtual machines to orchestration and reporting.
- Puppet ensures consistency, reliability and stability. It also facilitates closer collaboration between system admins and developers, enabling more efficient delivery of cleaner, better-designed code.

How puppet works:

- Once you install Puppet, every node (physical server, device or virtual machine) in your infrastructure has a Puppet agent installed on it. You'll also have a server designated as the Puppet master.
- Enforcement takes place during regular Puppet runs, which follow these steps:
    - Fact collection. The Puppet agent on each node sends facts about the node's configuration — detailing the hardware, operating system, package versions and other information — to the Puppet master.
    - Catalog compilation. The Puppet master uses facts provided by the agents to compile detailed data about how each node should be configured — called the catalog — and sends it back to the Puppet agent.
    - Enforcement. The agent makes any needed changes to enforce the node's desired state.
    - Report. Each Puppet agent sends a report back to the Puppet master, indicating any changes that have been made to its node's configuration.
    - Report sharing. Puppet's open API can send data to third-party tools, so you can share infrastructure information with other teams.

Puppet Architecture:





Configuration Language:

- "Puppet's configuration language has always been focused on the best combination of simplicity and power, and my goal was always to have it be more like a configuration file than a programming language," wrote Luke Kanies, founder and CEO of Puppet Lab.
- It supports DSL (domain specific language).

Transaction:

- Once the catalog is entirely constructed, it is passed on to the Transaction
- Transaction runs on the client, which pulls the Catalog down via HTTP
- The transaction performs a relatively straightforward task: walk the graph

  the order specified by the various relationships, and make sure each resource is in sync.

Resource Abstraction Layer:

- The work is actually done by the Resource Abstraction Layer (RAL), The RAL was the first component created in Puppet, it most clearly defines what the user can do.
- The job of the RAL is to define what it means to be a resource and how resources can get work done on the system.


```
Puppet::Type.newtype(:file) do
    ...
    newproperty(:content) do
        def retrieve
            File.read(@resource[:name])
        end
        def sync
            File.open(@resource[:name], "w") { |f| f.print @resource[:content] }
        end
    end
end
```

Day 1 – Installation & Configuration:

Pre- requisites:

- Install VMware Workstation 10x & Cent OS 6.x in your laptop.
- High speed Internet
- LINUX commands & VIM Editor with VMware Knowledge.

Set up the hostname (do this in agent also, hostname = agent.example.com)

Cat /etc/sysconfig/network
Vim /etc/sysconfig/network
Hostname = Server1.example.com
Hostname server1.example.com

Getenforce

The above command "getenforce" reports whether SELinux is enforcing, permissive, or dis-abled.

Iptables -F

It's a linux based firewall controlled by the program. The above command will delete all the rules in Iptables

Service iptables off

It will stop the iptables services.

Chkconfig iptables off

It will disable the enabled firewall

Selinux (or sestatus)

It will display the selinux status whether its enabled or disabled.

https://fedoraproject.org/wiki/EPEL

Wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

http://yum.puppetlabs.com/

wget http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm

The above command will download the necessary packages.

we need to copy this packages to agent (find agent ip first using ifconfig)

ls

Scp epel-release-6-8.noarch.rpm **agent-ip**:/root (agent password)

Copying the packages to agent machine.

Scp puppetlabs-release-el-6.noarch.rpm **agentip**:/root

Copying the packages to agent machine.

Rpm –ivh epel-release-6-8.noarch.rpm

Extracting the package.

Rpm –ivh puppetabs-release-el-6.noarch.rpm

Extracting the package.

Yum install puppet puppet-server facter

Installing the puppet server in Master machine

In agent (ssh root@agent-ip)

Yum install puppet facter –y

Installing the puppet in agent machine.

In master

Service puppetmaster start

Netstat –tupnl |grep 8140

Starting the Puppet services in master server.

In agent

Puppet agent –test –server=server1.example.com

The above command will sync the agent with Master server. It should execute only in agent machine.

In master -> Puppet cert –list (listing certificates)

In master -> Puppet cert sign agent.example.com (accepting agent certificates) (if there are 1000 nodes, master hast to accept these 1000 csr)

In agent - > Puppet agent –test –server=server1.example.com

To automate the CSR

From master

Puppet cert clean agent.example.com (to remove or unregister an agent's certificate)

To list all nodes that are signed or nonsigned

Puppet cert -list –all (+sign means it is signed)

run this on agent to remove previous master associations

Rm –rf /var/lib/puppet/ssl/*

How to enable automatic signing of CSR in master

Puppet config print| grep autosign

//Autosign=/etc/puppet/autosign.conf

Touch /etc/puppet/autosign.conf (touch – blank file is created)

Echo "*" > /etc/puppet/autosign.conf

Echo "*.example.com" > /etc/puppet/autosign.conf

(Puppet only operates on domain names, not on ip addresses)

In agent

Puppet agent –test –server=server1.example.com

Vim etc/puppet/puppet.conf and type

[agent]
Server=server1.example.com
Runinterval=3600 (for every1 hour)
#.....
Save and Exit
Puppet agent –test
Cron job or use deamon to start puppet
Service puppet start (or cron job)
cPuppet config print | grep runinterval (every 30mins it will check)

In agent

Puppet cert –list --all
Puppet agent –t –debug

In server

Eg: open a file
File{ "tmp/file1":
Ensure=>file,
} // this will run **touch /tmp/file1** in nodes
Package{
'vim':
Ensure=>present,
} // equivalent to **yum install vim**

Resource – package and file
Title is vim, tmp/file1
Ensure is the attribute

Resource {'title':
Attribute=>value, }

Day one completed with Puppet installation, establishing connectivity between the master and the agent server.

Day 2 – How to assign classes/codes to nodes:

How to assign classes/codes to nodes?

Manifest path
In master
cd /etc/puppet/manifests/
Ll
Touch site.pp
Vim site.pp

```
node 'agent.examle.com'{
    file { '/tmp/my-test':
            ensure=>present,
            content=>"This is created by puppet\n",
    }
}
```

In agent
Puppet agent --test
Cat /tmp/my-test

Idempotancy (feature of puppet) – if any resource is available in the machine, puppet will not execute it unnecessarily .
Vim /tmp/my-test //and type something in the content and run puppet agent –test
Whatever changes we want to make in agent file, do it only through master.
Rule 1. Don't change the agent file directly.
Rm –rf /tmp/my-test (will remove agent file but then it creates it again according to puppet master)

In master site.pp file
Remove all content and type
Import 'linux/*.pp'
In master
Mkdir linux
Cd linux
Vim agent.pp
(/etc/puppet/manifest/linux/agent.pp)

```
node 'agent.example.com'
{
    file {"/tmp/my-test""
```

```
            ensure=>file,
            content =>  “this is separated by agent pp files\n”,
    }
}
```

In agent
Puppet agent –test –server= server1.example.com
Cat /tmp/my-test

Agent ➔ 8140 ➔ /etc/puppet/manifests/site.pp ➔
/etc/puppet/manifests/linux/*.pp

In master
ll linux/

how to create modules
in master
puppet config print | grep modulepath --color
cd /etc/puppet/modules (puppet reads modules from this directory)
puppet module generate dars-sshd (dars-sshd is my module name; dars is author name sshd is mdule)
mv dars-sshd sshd (renaming module and it is mandatory step)
ll (will list sshd also)
cd sshd
ll (will list manifest)
cd manifest
ll (will list init.pp)
vim init.pp (all lines are commented and end of the file class sshd is there,
class sshd{

} // we can create codes in this class. For eg:

```
class sshd{
    file{ “/tmp/my-test”:
    Ensure => file,
    content =>  “This is created by modules or class \n”,

}
} // this class can be used for n number for agents
```

In master
Vim /et/puppet/manifest/linux/agent.pp

```
Node ‘agent.exampe.com’
{
    Include sshd
```

}
Node ‘node1.example.com’{
Include sshd
}
In agent
Puppet agent –test
Cat /tmp/my-test

In agent
Right click on agent removal devices

Vm – settings - >cd/dvd->use iso->browse (2 selected
Open terminal in agent
Cd /media/Centos_6.5Final/
Vim /etc/yum.repose.d/local.repo and type
[local]
Name=local
Baseurl=file://media/CentOS_6.5_Final/
Gpgcheck=0
Enabled=1
Cat /etc/yum.repose.d/local.repo and type

Yum clean all
Yum install
Mv /etc/yum.repose.d/epel* /root/
Mv /etc/yum.repose.d/puppetlabs.repo /root/
Mv /etc/yum.repose.d/ CentOS-* /root/
Ll /etc/yum.repose.d
Yum clean all
Yum list

Automating the ssh server
Go to master
3 steps to configure ssh
//we need package openssh-server, file : /etc/ssh/sshd_config, service :sshd

Cd /etc/puppet/modules/sshd/manifests/
Vim init.pp
class sshd {
    package { ‘ssh-pkg’:
            Name => ‘openssh-server’,
Ensure => ‘installed’,
}
File { ‘ssh-file’:

Name => /etc/ssh/sshd_config',
Ensure => 'present',
Source => 'puppet:///modules/sshd/sshd_sample', //this statement is added later
}
Service {'ssh-service':
Name => 'sshd',
Ensure => 'running',
Enable => 'true',
}
}
Puppet agent –test
Cat /etc/ssh/sshd_config //file is blank so add content in file {'ssh-file
In server
Vim /etc/…/sshd/sshd_sample
Write some content
In server we need to create directory files in sshd
Mkdir files
Cd files (puppet://modules.sshd.sshd_sample)
Path is /etc/puppet/modules/sshd/files
Cp /etc/ssh/sshd_config /etc/puppet/modules/sshd/files/sshd_sample

On agent
Cat /etc/ssh/sshd_config
Puppet agent –test
In server change permission
Chmod 777 sshd_config
In agent
Cat /etc/ssh/sshd_config (shows 100s of lines)

If we remove the sshd package from agent, it will not work, because we are not specifying any relationships. If the package is not there, file will not be there, and no ile means no service. Ie. Execute the package first then only files. Service is dependent on files and files is dependent on package. For relationships, puppet is providing parameters – Metaparameters of relationship

Metaparameters are

- require
- Before - assign priorities
- Notify – update target resource
- subscribe

in class sshd, packages -> write **before => File['ssh-file'],**

in file {}

require => Package[‘ssh-pkg’],

notify => Service[‘ssh-service’],

in service{}

subscribe => File[‘ssh-file’],

in agent

rm –rf /etc/ssh/sshd_onfig.rpmsave

service sshd status

yum remove openssh (just fo testing)

puppet agent –test

netstat –tupnl 22

How puppet can filter the type of agents

How to filter the fats

In agent

Facter

Facter | grep –color osfamily

Any facts can be filtered. Eg: osfamily, bios, etc..

There are three filters

1. selector
2. case statement
3. if else

in master

vim /etc/puppet/modules/sshd/manifests/init.pp

in class sshd {}, we are giving dynamic values.

Class sshd {

$sshpkg = $osfamily ? {

```
'RedHat' => 'openssh-server',

'Debian' => 'ssh',,

'Solaris' => 'ssh',

Default =>undef,

 }

$sshconfig = $osfamily ? {

'RedHat' => '/etc/ssh/sshd_config',

'Debian' => '/etc/ssh/sshd_config',

'Solaris' => '/etc/sshd_config',

Default => undef,

}

$sshservice =$sfamily ? {

'RedHat' => 'sshd',

'Debian' => 'ssh',

'Solaris' => 'ssh',

Default => undef,

}

            package { 'ssh-pkg':
                        Name => "$sshpkg",
                        Ensure => 'installed',
            }
            File { 'ssh-file':
                Name => "$sshconfig",
                Ensure =>  'present',
                Source => 'puppet:///modules/sshd/sshd_sample',         //this statement is
            added later
            }
            Service {'ssh-service':
                Name => "$sshservice",
                Ensure => 'running',
                Enable => 'true',
```

```
        }
}
}
```

In agent

Puppet agent –test

//yum remove openssh-server

In init.pp

Before package comment all dynamic variables facts and type

```
Case $::osfamly{
        'RedHat':{ $sshpkg = 'openssh-server'}
        'Debian': {$sshpkg = 'ssh'}
        'Solaris' :{ $sshpkg='ssh'}
Default: { }
}
Case $::osfamly{
        'RedHat':{ $sshconfig = '/etc/ssh/sshd_config'}
        'Debian': {$sshcofig = '/etc/ssh/sshd_config'}
        'Solaris' :{ $sshconfig='/etc/ssh_config'}
Default: {}
}
Case $::osfamly{
        'RedHat':{ $sshservice= 'sshd'}
        'Debian': {$sshservice = 'ssh'}
        'Solaris' :{ $sshservice='ssh'}
Default: {}
}
```

Or using if

```
Class sshd{
{
        If $::osfamily == 'Debian' {
                $sshpkg= 'ssh'
        }
        Elseif $::osfamily == 'RedHat'{
        $sshpkg = 'openssh-server-
        }
        Else
        {
                notify{   osfamily $::osfamily not found }
        }
}
```

How to create parameterized classes

In module folder

Puppet module generate dars-httpd

mv dars-sshd sshd (renaming module and it is mandatory step)
ll (will list sshd also)
cd sshd
ll (will list manifest)
cd manifest
ll (will list init.pp)

vim init.pp

```
Class httpd{
        class { '::httpd::install':} or class { 'httpd::install':}
        class { '::httpd::config':}
```

```
        class { '::httpd::service':}
}
Touch install.pp
Touch config.pp
Touch service.pp
Vim install.pp
Class httpd::install{
Package{ 'web-pkg':
        Ensure => 'installed',
        Name => 'httpd',
        Before => Class["httpd::config"],
}
}
Vim config.pp
Class httpd::config{
file{ 'web-config':
        ensure => 'present',
        name => '/etc/httpd/conf/httpd.conf',
                //or name => '/etc/httpd/conf.d/first.conf',  (virtual hosting)
                // source => 'puppet:///modules/httpd/first_sample',
        require => Class["httpd::install"],
        notify => Class["httpd::service"],
}
File { "/var/www/first":
Ensure => directory,
}
```

```
        file { “index.html”:                        //creating webpage in agent
                ensure => ‘present’,
                name => ‘/var/www/html/index.html’,
                source => ‘puppet:///modules/httpd/index_sample’,
        }

}
```

Vim service.pp

```
Class httpd::service{
service{ ‘web-service’:
        Name => ‘httpd’,
        Ensure => ‘running’,
        Enable => ‘true’,
        Subscribe => Class[“httpd::config”],
}
}
```

Ll

Vim /etc/puppet/manifest/linux/agent.pp

Change **include sshd** to **include httpd**

In agent

Puppet agent –test

Virtual hosting (check Class httpd::config)

Mkdir files (under /etc/puppet/modules/httpd)

Cd files

Touch first_sample

Touch index_sample

Vim index_sample

Write some content

Vim first_sample

<VirtualHost *:80>
ServerAdmin root@first.conf
ServerName first.com
DocumentRoot /var/www/first
</VirtualHost>

…..

Firefox …

Service httpd restart

Definitions - to have multiple virtual instances

In class we cannot

Backup1

At 12

Back up

At 3:00

File { “job1”

Name=>backup1

}

Cron {12:00}

File { “job2”

Name=>backup2

}

Cron {12:00}

IN MASTER (etc/puppet/modules/httpd/manifests.)

Vim website.pp

define: httpd::website {

```
        incude httpd

        file {“etc/httpd/conf.d/$(site_name).conf”:

        content => template(‘httpd/vhost.conf.erb’)        // not using source key word.
Source is for static value

        notify => Class[‘httpd::service’],

        }

        File {“/var/www/${site_name}”:

                Ensure=>directory

        }

        File {“/var/www/${site_name}/index.html”:

        Ensure => file,

        Source => “puppet:///modules/httpd/${site_name}”,

        }

}
```

Vim init.pp

Write new module name

```
httpd::website{‘linux-links’:
        site_domain=>’linux-links.com’,}
//$name =linux-links
```

vim website.pp

change define httpd… to

```
define httpd::website($site_domain){
        include httpd
        $site_name =$name     /linux-links.com
```

In module
Mkdir templates
Cd templates
Vim vhost.cnf.erb (erb means extended ruby)

```
<VirtualHost *:80>
        ServerName <%= site_domain %>
        DocumentRoot /var/www/<%= site_name %>
        ServerAdmin admin@<%= site_domain %>
```

</VirtualHost>
Vim linux-links

Cat /etc/httpd/conf.d/linux-links.conf

Now move to manifest in httpd module
Vim init.pp
…..

In agent -> Uncomment namevirtual host to enable multiple domains -
/etc/httpd/conf/httpd.conf
In server Edit /etc/hosts
In agent firefox ….com

Day 3 – User Management / SSh Keygen:

Users Management

In Master

1. Create module user

cd /etc/puppet/modules
puppet module generate username-user
mv username-user user

2. Puppet describe resource user

a. Get detailed information about various users

3. Puppet resource user root

a. Displays the information about the user

4. Go to modules/users/manifests/init.pp

```
class users{
user {admin:
ensure => 'present',
comment => 'Administrator',
gid => '1020',
home => '/home/admin/',
managehome => '/home/admin/',
password => '',
password_max_age => '9999',
password_min_age => '0',
shell => '/bin/bash',
uid => '1001',
```

require => Groupd['ops'],
}
group {'ops':
gid => '1020',
ensure => 'present,
}
}

5. vim /etc/puppet/manifests/linux/agent.pp

node 'agent.example.com'{
include httpd
include users }

6. execute the following command, enter detault password (desired) and copy the hash obtained:

openssl passwd -1

7. insert the password hash in the user module for the particular user in the password section

In agent
puppet agent --t --server=server.example.com
How to manage SSH Keys using puppet
In master
ssh-keygen
ssh-copy-id -I /root/.ssh/id_rsa.pub admin@<ipaddress>
ssh admin@<ipaddress>
cat /root/.ssh/id_rsa.pub

8. exclude ssh-rsa, server details and copy the has from the id_rsa.pub

9. insert the following in the init.pp in module users

ssh_authorized_key { 'masterkeys':
user => 'admin',
type => 'rsa',
key => 'haskeykljasd;fkja;llskdjf;llaskdjfp",
}

10. code snippet to execute a specific linux command using puppet

exec { 'Download the data':
cwd => '/tmp/,
command => '/usr/bin/wget http://www.linux-links.com/puppet.tar.gz',
creates => 'tmp/puppet.tar.gz', #checks if the file already exists to avoid repeated downloads
}

How to manage Cron jobs using Puppet

cron { 'Backup the data':
command => '/usr/bin/rsync -az /var/www/html /home/',
hour => '04'
minute => '00',
}

Managing Environments
Environments

- Production- default environment

o /etc/puppet/modules/

o /puppet/manifests/

- Dev- development environment

Check current environment
puppet config print | grep environment
Information is in the directory : /etc/puppet/environments
Set Environment path in Puppet master
vim /etc/puppet/environments
environmentpath = /etc/puppet/environments
cd /etc/puppet/environments/
mkdir dev
cd dev
mkdir modules
mkdir manifests
touch manifests/site.pp
vim environment.conf
Insert the following lines in the file environment.conf
[dev]
modulepath = /etc/puppet/environments/dev/modules
manifest = /etc/puppet/environments/manifests/site.pp
The following lines in the file site.pp in manifest directory
node 'agent.example.com' {
file { "/tmp/env":
ensure => 'file',
content => "this istesting for env \n",
}
}
In the agent
puppet agent –t –environment=dev
In master
service puppet master start #so the changes are reflected
In the agent
vim /etc/puppet/puppet.conf
change environment = dev

Server Provisioning using Puppet

Foreman
An open source tool we use with puppet for Server provisioning, we use TFTP, DHCP and DNS for server provisioning. Foreman has a GUI which we can use.
Site: TheForeman.org
Requirements & Configuration

▫ TFTP Protocol - Trivial File Transfer Protocol

o Installation

▫ Go to yum.theforeman.org and download theforeman rpm file, run:

▫ *rpm -ivh <installationfile>.rpm*

▫ *yum install foreman-installer*

o stop puppet master and start foreman installer

▫ service puppetmaster stop

▫ foreman-installer

▫ use the random password shown and loginto theforeman UI

▫ change foreman password

▫ import environments from the puppet master through the UI

o Config Groups

▫ Import

o Sign agent with Foreman (repeat the agent signing with puppet again)

▫ Goto agent

▫ Clean certificates

▫ Goto master

▫ Clean certificates

▫ Re-associate/ Sign agent with master /

▫ Go back to foreman and it should be visible

o In agent mount the installation media for the OS

▫ Installation media

o Required for provisioning

- DHCP

o Configuration done during the Foreman configuration procedures

Reports
Find whether the report is enabled or not

- *puppet config print | grep report*

Find the report directory

- *puppet config print | grep reportdir*

Get detailed debug information

- *puppet agent --test --debug*

Get detailed summary information

- *puppet agent --test --summarize*

Look for any changes in Puppet Master

- *puppet agent --test --noop*

Validate any bugs or errors in the manifest files

- *puppet parser validate manifests/nodes.pp*

Search for a specific module in the puppet forge online

- *puppet module search <module-name>*

Install a specific module from online puppet forge

- *puppet module install <module-name>*

Search for a specific module in the puppet forge online

- *puppet module search <module-name>*

Differences between Puppet Enterprise & Open Source

Open Source	**Enterprise**
Foreman UI	Built in UI
Integrated Vcentre compatibility	
/etc/puppet/*	/etc/puppetlabs/*
/var/lib/puppet/*	/etc/puppetlabs/*

Other useful tools related to Puppet

- mCollective - Inventory gathering tool
- Hiera – key/value lookup tool for configuration data
- PuppetDB – Collects data generated by Puppet, like reports, etc.