Adding a resource to a node

node default {

package { 'httpd':

ensure => 'installed'

}

}

- Resource Abstraction Layer (RAL)

- Các package được biên dịch tới các system khác, provider được sử dụng để xác thực các gói tín có được puppet hỗ trợ cho việc cài đặt. vd: Yum provider, APT provider..

Using Facter to describe a node

- Facter tách riêng các nền tảng khác nhau trong Puppet, sử dụng cho việc đồng bộ các hệ điều hành khác nhau

- Có 1 lượng các facter cài đặt mặc định hỗ trợ việc sử dụng puppet

Variables- Sử dụng dấu “$”, sử dụng trong 1 manifest theo dạng “${variable}” thay vì $variable. Theo dạng fully qualified domain name (FQDN) gọi theo dạng “${::fqdn}” (sử dụng theo cấu trúc cây puppet trong 1 module)

Scope- chính là ký tự “::”, sử dụng trong các biến ${::test::a} ${::test::b} để tạo ra cấu trúc cây, phân biệt các tên biến giống nhau nhưng nằm trong các class khác nhau.

Cài packet trước khi chạy

service {'httpd':

ensure => running,

require => Package['httpd'],

}

package {'httpd':

ensure => 'installed',

}

Capitalization(Đóng gói tên biến, class, file ..)

- Đóng gói các class, file … sử dụng chúng cho việc định nghĩa các phụ thuộc

- VD: Package['httpd'], Class['something::somewhere'], đối với type đc định nghĩa “example::thing {'one':}” => gọi bằng “Example::Thing['one']”

Learning metaparameters and ordering (Sử dụng định nghĩa sự phụ thuộc)

- Đối với các phụ thuộc, vd: cài trước khi chạy, .. thì cần tới các metapara.

- Các meta được cung cấp:

+ before

+ require

+ notify

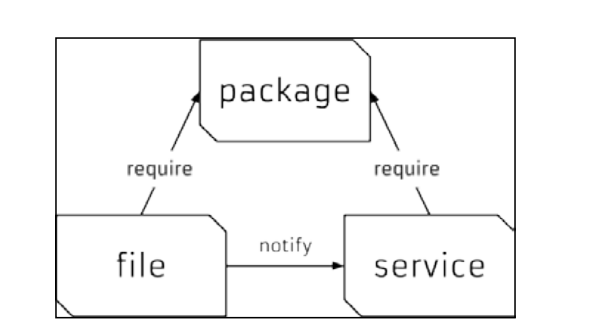
+ subscribe

- before, require chỉ định thứ tự. notify thuộc before, subscribe thuộc require.

- notify sử dụng cho các service, chắc chắn service phải restart trước khi apply các resource mới. Được đĩnh nghĩa trên các file resource thêm vào

- subcribe thì ngược lại, subcribe được định nghĩa trên service, service quyết định restart lại khi nào

Trifecta- quan hệ giữa notify và require



Idempotency- 1 khái niệm chính của puppet, khi node apply catalog, nó sẽ ko gây ảnh hưởng tới puppet chạy. => dù có bất kỳ thay đổi nào sẽ vẫn dẫn đến 1 trạng thái biết trước. Nếu xảy ra lỗi trong quá trình apply catalog => quá trình sẽ bị hủy bỏ

Installing, configuring, and starting a service

VD: cài đặt theo kịch bản:

+ cài đặt httpd => add file cấu hình => restart service

# chắc chắn service phải chạy, yêu cầu package

service {'httpd':

ensure => running,

require => Package['httpd'],

}

# định nghĩa package sẽ được sử dụng

package {'httpd':

ensure => installed,

}

# cài file cấu hình, chắc chắn package đã cài, restart service sau khi chạy

file {'/etc/httpd/conf.d/cookbook.conf':

content => "<VirtualHost \*:80>\nServername

cookbook\nDocumentRoot

/var/www/cookbook\n</VirtualHost>\n",

require => Package['httpd'],

notify => Service['httpd'],

}

# tạo file index theo đường dẫn của file cấu hình, yêu cầu đường dẫn tồn tại

file {'/var/www/cookbook':

ensure => directory,

}

file {'/var/www/cookbook/index.html':

content => "<html><h1>Hello World!</h1></html>\n",

require => File['/var/www/cookbook'],

}

Các luật trong ví dụ trên

+ httpd will be running

+ The VirtualHost configuration file will exist

+ httpd will restart and be aware of the VirtualHost file

+ The DocumentRoot directory will exist

+ An index.html file will exist in the DocumentRoot directory

Creating a manifest

- Để test 1 manifest sử dụng cmd “puppet apply site.pp”

VD: Tạo file manifest site.pp trong thư mục ../code/env../manifests/site.pp

node default {

file { '/tmp/hello':

content => "Hello, world!\n",

}

}  
\_\_\_

#apply site.pp

Checking your manifests with Puppet-lint- 1 số chuẩn code trong puppet, làm code dễ học, dễ hiểu, nâng cấp

+ Must use two-space soft tabs

+ Must not use literal tab characters

+ Must not contain trailing white space

+ Should not exceed an 80 character line width

+ Should align parameter arrows (=>) within blocks

- Để sử dụng, đầu tiên install package “puppet-line”

package {'puppet-lint':

ensure => 'installed',

provider => 'gem',

}

# puppet apply puppet-lint.pp

- Để check cú pháp, sử dụng câu lệnh “puppet-lint site.pp”

Using modules

- Định nghĩa modules (đọc trong các tl trước)

- Để tạo modules sử dụng cmd “puppet module generate lacoski-test”

- Các định nghĩa module sẽ trong “[Tên modules]/manifests/init.pp”

- Các sources sẽ được đặt trong “[Tên modules]/files/[tên..]” khi truy vấn thì sẽ chỉ “[Tên modules]/[tên..]”

VD:

file { '/etc/memcached.conf':

source => 'puppet:///modules/memcached/memcached.conf',

}

\_\_\_\_\_

MODULEPATH/ (/home/thomas/.puppet/modules)

└memcached/

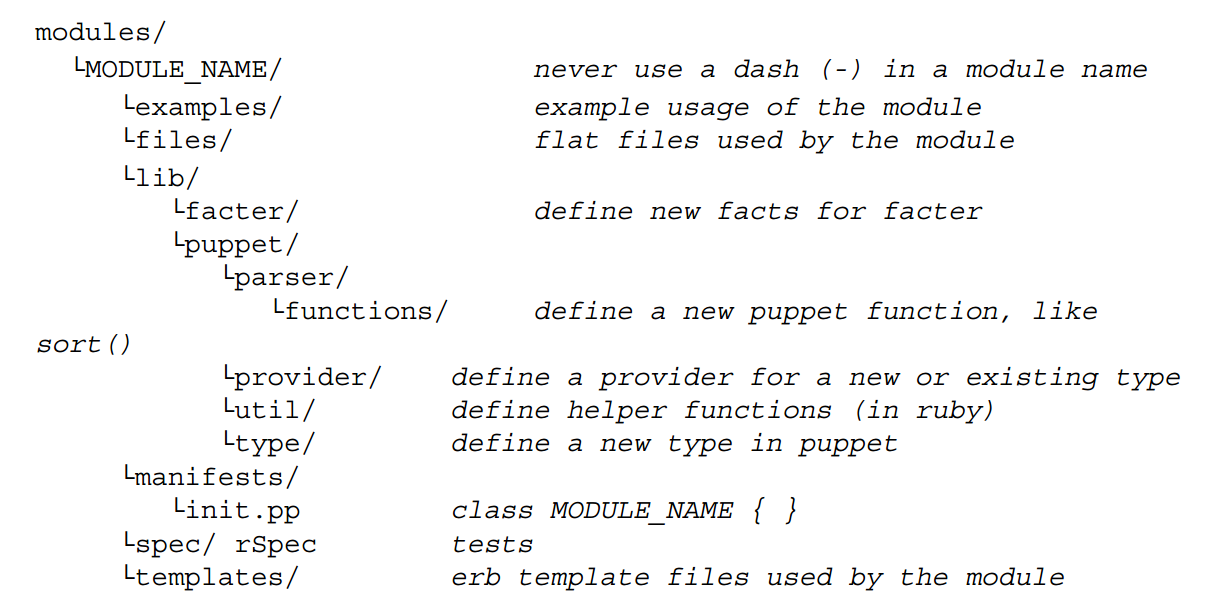
└files/

└memcached.conf

- Để test module sử dụng cmd: “puppet apply --modulepath=/home/thomas/.puppet/

modules /home/thomas/.puppet/manifests/site.pp”

- Cấu trúc thường có của 1 module, ko nhất thiết 1 số path sẽ tồn tại



Templates

- Tìm hiểu thêm

VD:

file { '/etc/memcached.conf':

content => template('memcached/memcached.conf.erb'),

}

MODULEPATH/memcached/templates/memcached.conf.erb

Faces, function, types, providers

- Tìm hiểu thêm

Using standard naming conventions- Đặt trên module liên quan đến service nó quản trị

- Các chức năng của class sẽ thuộc subclass. VD: apache::vhosts or rails::dependencies.

- Đặt tên chức năng class ngắn gọn, xúc tích

- Trong môi trường làm việc nhóm, các file cấu hình có dạng: setup.thanh.conf, setup.hoang.conf => phân biệt các cấu hình của từng người

- Sử dụng file config theo release

VD:  
source = > "puppet:///modules/memcached

/memcached.${::lsbdistrelease}.conf",

Using inline templates- Template sử dụng Embedded Ruby (ERB) giúp cho việc các config file động

VD:  
cron { 'chkrootkit':

command => '/usr/sbin/chkrootkit >

/var/log/chkrootkit.log 2>&1',

hour => inline\_template('<%= @hostname.sum % 24 %>'),

minute => '00',

}

Iterating over multiple items- Sử dụng array cho việc lặp lại

VD:

$packages = [ 'ruby1.8-dev',

'ruby1.8',

'ri1.8',

'rdoc1.8',

'irb1.8',

'libreadline-ruby1.8',

'libruby1.8',

'libopenssl-ruby' ]

package { $packages: ensure => installed }

Using hashes

- Tính chất giống array, nhưng truy vấn theo tên, nó cực hưu dụng khi muống gán các giá trị valu, string, function, sử dụng từ khóa để gọi

$interface = {

'name' => 'eth0',

'ip' => '192.168.0.1',

'mac' => '52:54:00:4a:60:07'

}

notify { "(${interface['ip']}) at ${interface['mac']} on

${interface['name']}": }

Creating arrays with the split function

- Sử dụng cho việc tách riêng chức năng lặp

VD:

define lunchprint() {

notify { "Lunch included ${name}":}": }

}

$lunch = ['egg', 'beans', 'chips']

lunchprint { $lunch: }  
#### kq

t@mylaptop ~ $ puppet apply lunchprint.pp

...

Notice: Lunch included chips

Notice: Lunch included beans

Notice: Lunch included egg

- Sử dụng split function cho việc chia array. Cấu trúc: split($array, ‘ký tự chia’)

VD:

$menu = 'egg beans chips'

$items = split($menu, ' ')

lunchprint { $items: }

#### KQ

t@mylaptop ~ $ puppet apply lunchprint2.pp

...

Notice: Lunch included chips

Notice: Lunch included beans

Notice: Lunch included egg.

Cấu trúc điều kiện

- Sử dụng tương tự như các ngôn ngữ lặp trình khác

- Các ký tự hỗ trợ “==, !=, >, <=,”

- Có thể kết hợp biểu thức “and, or”

VD:

if $::timezone == 'UTC' {

notify { 'Universal Time Coordinated':}

} else {

notify { "$::timezone is not UTC": }

}

\_\_

if ($role == 'webserver') and ( ($datacenter == 'A') or ($datacenter

== 'B') ) {

…

}

Using regular expressions in if statements- Tìm hiểu thêm

Using selectors and case statements- Tương tự if nhưng nó cung cấp form, cấu trúc đơn gian

VD:

$systemtype = $::operatingsystem ? {

'Ubuntu' => 'debianlike',

'Debian' => 'debianlike',

'RedHat' => 'redhatlike',

'Fedora' => 'redhatlike',

'CentOS' => 'redhatlike',

default => 'unknown',

}

notify { "You have a ${systemtype} system": }

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class debianlike {

notify { 'Special manifest for Debian-like systems': }

}

class redhatlike {

notify { 'Special manifest for RedHat-like systems': }

}

case $::operatingsystem {

'Ubuntu',

'Debian': {

include debianlike

},

'RedHat',

'Fedora',

'CentOS',

'Springdale': {

include redhatlike

}

default: {

notify { "I don't know what kind of system you have!":

}

}

}

- Cấu trúc select giống biểu thực điều kiện loại 3, biểu thức case thì tương tự ngôn ngữ lập trình

Defaults:

- Các select và case để có biểu thức giá trị mặc định

$lunch = 'Filet mignon.'

$lunchtype = $lunch ? {

/fries/ => 'unhealthy',

/salad/ => 'healthy',

default => 'unknown',

}

notify { "Your lunch was ${lunchtype}": }

NOTE:

- When the default action shouldn't normally occur, use the fail() function to halt the

Puppet run.

Using the in operator- Sử dụng để tìm 1 substring trong string, tìm 1 phần tử trong array

VD:

##

if 'spring' in 'springfield'

##

if $crewmember in ['Frank', 'Dave', 'HAL' ]

##

$ifaces = { 'lo' => '127.0.0.1',

'eth0' => '192.168.0.1' }

if 'eth0' in $ifaces {

notify { "eth0 has address ${ifaces['eth0']}": }

}

VD:

if $::operatingsystem in [ 'Ubuntu', 'Debian' ] {

notify { 'Debian-type operating system detected': }

} elseif $::operatingsystem in [ 'RedHat', 'Fedora', 'SuSE',

'CentOS' ] {

notify { 'RedHat-type operating system detected': }

} else {

notify { 'Some other operating system detected': }

}

- “in” opera hỗ trợ boolean, cho phép gán trực tiếp cho 1 biến

$debianlike = $::operatingsystem in [ 'Debian', 'Ubuntu' ]

if $debianlike {

notify { 'You are in a maze of twisty little packages, all alike': }

}

Lambda functions- Hỗ trợ các each, map, filter, reduce, or slice

- Tìm hiểu thêm

Puppet Infrastructure

- Để cài đặt Puppet, xem tài liệu triển khai

- Để hiện thị version puppet sử dụng cmd: “#:~ puppet --version”

Managing your manifests with Git- Quản trị modules, manifest với git. Tạo 1 private git repo, sau đó clone nó tới modules directory trong puppet

VD:

Bước 1: cài git

package {'git':

ensure => installed

}

Bước 2: apply git

#puppet apply git.pp

Bước 3: Tạo user, group

group { 'git':

gid => 1111,

}

user {'git':

uid => 1111,

gid => 1111,

comment => 'Git User',

home => '/home/git',

require => Group['git'],

}

file {'/home/git':

ensure => 'directory',

owner => 1111,

group => 1111,

require => User['git'],

}

file {'/home/git/repos':

ensure => 'directory',

owner => 1111,

group => 1111,

require => File['/home/git']

}

Bước 4: Truy cập vào git user, tạo kho chứa, tạo pass

# sudo -iu git

git@git $ cd repos

git@git $ git init --bare puppet.git

Initialized empty Git repository in /home/git/repos/puppet.git/

# passwd git

Bước 5: tạo chứng chỉ ssh truy cập giữa local vs git private repo

Bước 6: clone repo với local, nơi muốn lưu trữ các module (../code/..)

t@mylaptop ~$ git clone git@git.example.com:repos/puppet.git

Cloning into 'puppet'...

warning: You appear to have cloned an empty repository.

Checking connectivity... done.

Bước 7: setup cơ bản git

Bước 8: Từ giờ đã có thể commit lên git server

Writing a papply script- Viết đoạn script apply nhanh 1 số câu lệnh

VD:

Bước 1: Tạo module

Bước 2: Tạo script modules/puppet/files/papply.sh

#!/bin/sh

sudo puppet apply /etc/puppet/cookbook/manifests/site.pp \

--modulepath=/etc/puppet/cookbook/modules $\*

Bước 3: Chỉnh sửa file ../manifests/init.pp

class puppet {

file { '/usr/local/bin/papply':

source => 'puppet:///modules/puppet/papply.sh',

mode => '0755',

}

}

Bước 4: Chỉnh sửa file ../manifests/site.pp

node default {

include base

include puppet

}

Bước 5: Apply this modules

puppet apply /etc/puppet/cookbook/manifests/site.

pp --modulepath /etc/puppet/cookbook/modules

Bước 6: Đã thể test script

- Có thể dùng cron để tự động chạy script

VD:

cron { 'run-puppet':

ensure => 'present',

command => '/usr/local/bin/papply,

minute => '\*/10',

hour => '\*',

}

Creating a centralized Puppet infrastructure- Puppet là công cụ rất mạnh trong việc quản lý tập trung lượng lớn các node, nếu các node < 700 => puppet master có thể giải quyết được hết. Nếu số lượng lớn hơn => phải dùng kỹ thuật load balancing để bảo đảm hiệu năng của node master

- Puppet defaule sử dụng X509 certificate authority và distributes catalogs (compiled manifests). Công nghệ puppet được xây dựng trên WEBrick,

- Puppet sử dụng package puppet master

Creating certificates with multiple DNS names

- Bình thường việc xác nhận node agent diễn ra thủ công => tại phần này ta sẽ tạo chứng chỉ cho phép puppet master xác nhận nhiều node

- Đề làm được điều này, puppet master cần có thêm certificate authority (CA).

VD:

Bước 1: Dừng puppet master

# service puppetmaster stop

[ ok ] Stopping puppet master

Bước 2: Xóa các chứng chỉ hiện có

# puppet cert clean puppet

Bước 3: Tạo puppet certificate với tham số --dnsalt-names

root@puppet:~# puppet certificate generate puppet --dns-alt-names

puppet.example.com,puppet.example.org,puppet.example.net --calocation

local

Notice: puppet has a waiting certificate request

True

Bước 4: ký trước các chứng chỉ vừa tạo

root@puppet:~# puppet cert --allow-dns-alt-names sign puppet

Notice: Signed certificate request for puppet

Notice: Removing file Puppet::SSL::CertificateRequest puppet at '/

var/lib/puppet/ssl/ca/requests/puppet.pem'

Bước 5: Restart service Puppet master

- Từ bây giờ, khi puppet agent xin chứng chỉ từ puppet master, master sẽ check các chứng chĩ đã tạo trước, nếu khớp sẽ cấp xuống cho agent

Running Puppet from passenger- WEBrick server không phù hợp với cho việc xử lý lượng lớn các node => giải pháp là sử dụng **Passenger**

**-** Để install puppetmaster-passenger

#puppet resource package puppetmaster-passenger ensure=installed

- Các bước setup

Bước 1: Setup các file apache config, tùy thuộc vào nền tảng “/etc/httpd/conf.d or /etc/apache2/sitesenabled”

PassengerHighPerformance on

PassengerMaxPoolSize 12

PassengerPoolIdleTime 1500

# PassengerMaxRequests 1000

PassengerStatThrottleRate 120

RackAutoDetect Off

RailsAutoDetect Off

Listen 8140

Bước 2: Add file virtualhost, theo port 8140

<VirtualHost \*:8140>

SSLEngine on

SSLProtocol ALL -SSLv2 -SSLv3

SSLCertificateFile /var/lib/puppet/ssl/certs/puppet.

pem

SSLCertificateKeyFile /var/lib/puppet/ssl/private\_keys/

puppet.pem

SSLCertificateChainFile /var/lib/puppet/ssl/certs/ca.pem

SSLCACertificateFile /var/lib/puppet/ssl/certs/ca.pem

SSLCARevocationFile /var/lib/puppet/ssl/ca/ca\_crl.pem

SSLVerifyClient optional

SSLVerifyDepth 1

SSLOptions +StdEnvVars +ExportCertData

Bước 3: 1 số headers được set để passenger process cho phép ssl infor gửi tới client node

RequestHeader unset X-Forwarded-For

RequestHeader set X-SSL-Subject %{SSL\_CLIENT\_S\_DN}e

RequestHeader set X-Client-DN %{SSL\_CLIENT\_S\_DN}e

RequestHeader set X-Client-Verify %{SSL\_CLIENT\_VERIFY}e

Bước 4: Main Config file là config.ru

DocumentRoot /usr/share/puppet/rack/puppetmasterd/public/

RackBaseURI /

Bước 5: config.ru phải tồn tại trong “/usr/share/puppet/rack/puppetmasterd/”

$0 = "master"

ARGV << "--rack"

ARGV << "--confdir" << "/etc/puppet"

ARGV << "--vardir" << "/var/lib/puppet"

require 'puppet/util/command\_line'

run Puppet::Util::CommandLine.new.execute

Bước 6: passenger apache đã được config tại config.ru, chạy service apache

root@puppet:~ # service apache2 start

- Từ bây giờ, passenger sử dụng các chứng chỉ Puppet master để listen port 8140 và xử lý các chứng chỉ SSL liên kết

Setting up the environment- mặc định env là production, có thể cấu hình thay thế tại agent config (puppet.conf) chỉnh giá trị environment = [tên giá trị]

- environmentpath chỉ định nơi các env được lưu

Configuring PuppetDB

Configuring Hiera- Cho phép quản lý các biến bên ngoài các manifest, quản lý phân cấp các giá trị, là giải pháp đối với nhưng cấu hình phức tạp

- Cài đặt Hiera (đọc docs)

VD:  
Bước 1: cấu trúc hiera là yaml file, /etc/puppet/hiera.yaml

:hierarchy:

- common

:backends:

- yaml

:yaml:

:datadir: '/etc/puppet/hieradata'

Bước 2: Tạo common.yaml được tham chiếu trong hierarchy

# ../hieradata/common.yaml

message: 'Default Message'

Bước 3: Tại site.pp tạo resource dựa theo hiera value

node default {

$message = hiera('message','unknown')

notify {"Message is $message":}

}

Bước 4: Chạy site.pp

- Hiera sử dụng hierarchy cho việc tìm kiếm tập các files, tìm giá trị của value

- Kiến trúc của hierarchy

:hierarchy:

- hosts/%{hostname}

- os/%{operatingsystem}

- network/%{network\_eth0}

- common

Writing Better Manifests

Introduction- Cách viết manifest file rõ ràng, dễ hiểu

Using arrays of resources- Sử dụng array cho list các resource

|  |  |
| --- | --- |
| package { 'sudo' : ensure => installed }  package { 'unzip' : ensure => installed }  package { 'locate' : ensure => installed }  package { 'lsof' : ensure => installed }  package { 'cron' : ensure => installed }  package { 'rubygems' : ensure => installed } | package  {  [ 'cron', 'locate', 'lsof', 'rubygems', 'sudo',  'unzip' ]:  ensure => installed,  } |

Using resource defaults- Puppet module là nhóm các resources có liên quan đến nhau, thường nó sử để định nghĩa 1 service chính xác.

VD: Đinh nghĩa apache service

Bước 1: Tạo apache module và tạo resource default

class apache {

File {

owner => 'apache',

group => 'apache',

mode => 0644,

}

}

Bước 2: Tạo html files trong /var/www/html

file {'/var/www/html/index.html':

content => "<html><body><h1><a

href='cookbook.html'>Cookbook!

</a></h1></body></html>\n",

}

file {'/var/www/html/cookbook.html':

content =>

"<html><body><h2>PacktPub</h2></body></html>\n",

}

Bước 3: Apply module

Using defined types- Trong trường hợp cần phải truyền vào parameter

VD:

# vim tmp.pp

define tmpfile() {

file { "/tmp/${name}":

content => "Hello, world\n",

}

}

tmpfile { ['a', 'b', 'c']: }

# puppet apply tmp.pp

- Mở rộng, trong vd trên ta chỉ sử dụng $name. Chúng ta có thể thêm 1 số parameter

VD:  
define tmpfile($greeting) {

file { "/tmp/${name}":

content => $greeting,

}

}

tmpfile{ 'foo':

greeting => "Good Morning\n",

}

- Chúng ta có thể định nghĩa nhiều parameter hơn

define webapp($domain,$path,$platform) {

...

}

webapp { 'mywizzoapp':

domain => 'mywizzoapp.com',

path => '/var/www/apps/mywizzoapp',

platform => 'Rails',

}

Note: Biến name ko cần định nghĩa

Using tags- Sử dụng khi cần biết chính xác 1 số trạng thái để áp dụng cho class, vd: service tường lửa cần biết chính xác port để mở, mà các node khác nhau có service khác nhau => mỗi node sẽ gắn 1 tag, service tường lửa sẽ phân loại các tag để có những áp dụng cụ thể

VD:

node 'cookbook' {

if tagged('cookbook') {

notify { 'tagged cookbook': }

}

}

VD:

## định nghĩa trong file site.pp

node 'cookbook' {

tag('tagging')

class {'tag\_test': }

}

## tại 1 manifest khác định nghĩa

class tag\_test {

if tagged('tagging') {

notify { 'containing node/class was tagged.': }

}

}

## vì class note ‘..’ có gắn tag(‘tagging’) => trong class sẽ chạy thêm 1 số tính năng khác

Using run stages- Khi cần xác nhận các resource nào cần được chạy trước, resource nào sẽ chạy sau. Nó là 1 loại metaparameter. Khi ta cần xác nhận 1 số class sẽ được chạy trước, 1 số class sẽ được chạy sau. Vd như first stage => main stage => last stage

VD: 2 class me\_first và me\_last được gắn trạng thái

class admin::stages {

stage { 'first': before => Stage['main'] }

stage { 'last': require => Stage['main'] }

class me\_first {

notify { 'This will be done first': }

}

class me\_last {

notify { 'This will be done last': }

}

class { 'me\_first':

stage => 'first',

}

class { 'me\_last':

stage => 'last',

}

}

- Chỉnh sửa site.pp

node 'cookbook' {

class {'first\_class': } # main stage

class {'second\_class': }

include admin::stage

}

- KQ:

root@cookbook:~# puppet agent -t

Info: Applying configuration version '1411019225'

Notice: This will be done first **#first stage**

Notice: Second Class #main

Notice: First Class #main

Notice: This will be done last **#last statge**

Notice: Finished catalog run in 0.43 seconds

Using roles and profiles- Tách riêng cách chức năng class thành nhiều role khác nhau

VD: Tách riêng web server thành role base và role apache => khiến cấu trúc code rõ ràng

$ vim profiles/manifests/base.pp

class profiles::base {

include base

}

$ vim profiles/manifests/apache.pp

class profiles::apache {

$apache = $::osfamily ? {

'RedHat' => 'httpd',

'Debian' => 'apache2',

}

service { "$apache":

enable => true,

ensure => true,

}

package { "$apache":

ensure => 'installed',

}

}

$ vim roles/manifests/webserver.pp

class roles::webserver {

include profiles::apache

include profiles::base

}

$ vim roles/manifests/webserver.pp

class roles::webserver {

include profiles::apache

include profiles::base

}

Passing parameters to classes- sử dụng khi cần giải quyết 1 số vấn đề phức tạp, vd: khi cần chỉ định version cho class => có thể truyền nó dưới dạng parameter

class eventmachine($version) {

package { 'eventmachine':

provider => gem,

ensure => $version,

}

}

## sử dụng

class { 'eventmachine':

version => '1.0.3',

}

VD:

class mysql($package, $socket, $port) {

…

}

class { 'mysql':

package => 'percona-server-server-5.5',

socket => '/var/run/mysqld/mysqld.sock',

port => '3306',

}

Passing parameters from HieraVD:

Bước 1: Tạo class mysql

##vim modules/mysql/manifests/init.pp

class mysql ( $port, $socket, $package ) {

notify {"Port: $port Socket: $socket Package: $package": }

}

Bước 2: update file .yaml trong hiera với các giá trị mặc định

---

mysql::port: 3306

mysql::package: 'mysql-server'

mysql::socket: '/var/lib/mysql/mysql.sock'

Bước 3: apply cho node và chạy

node default {

class {'mysql': }

}

# puppet agent -t

Writing reusable, cross-platform manifests- Sử dụng puppet cho việc quản trị nhiều nền tảng khác nhau

VD:

## phân biệt nền tảng

$ssh\_service = $::operatingsystem? {

/Ubuntu|Debian/ => 'ssh',

default => 'sshd',

}

service { $ssh\_service:

ensure => running,

}## phân biêt kiến trúc

$libdir = $::architecture ? {

/amd64|x86\_64/ => '/usr/lib64',

default => '/usr/lib',

}

Passing arguments to shell commandsVD:

Bước1: Tạo shellquote.pp

$source = 'Hello Jerry'

$target = 'Hello... Newman'

$argstring = shellquote($source, $target)

$command = "/bin/mv ${argstring}"

notify { $command: }

Bước 2: chạy puppet

# puppet apply shellquote.pp

Working with Files and Packages

- Cách móc nối các file với nhau, sinh file từ temple, install package từ 1 kho chứa ngoài

Making quick edits to config files

- Cách chỉnh sửa 1 hay nhiều phần nhỏ trong 1 file mà không cần sử thay đổi file cấu hình

- Cách đơn giản nhất là sử dụng file\_line type được cung cấp bởi puppetlabs-stdlib

- type này sẽ tự động add thêm dòng mới vào file

- cần phải có module “puppet module install puppetlabs-stdlib”

VD: add thêm 1 dòng mới vào file

file {'/tmp/cookbook':

ensure => 'file',

}

file\_line {'cookbook-hello':

path => '/tmp/cookbook',

line => 'Hello World!',

require => File['/tmp/cookbook'],

}

VD: chỉnh sửa 1 dòng

file {'/tmp/cookbook':

ensure => 'file',

}

file\_line {'cookbook-remove':

ensure => 'absent',

path => '/tmp/cookbook',

line => 'Hello World!',

require => File['/tmp/cookbook'],

}

file\_line {'cookbook-match':

path => '/tmp/cookbook',

line => 'Oh freddled gruntbuggly, thanks for all the

fish.',

match => 'fish.$',

require => File['/tmp/cookbook'],

}

Editing INI style files with puppetlabs-inifile- sử dụng cho việc config file puppet.conf (sinh ra dạng giống như file config)

- Trong inifile module định nghĩa 2 loại, ini\_setting và init\_subsetting.

- ini\_setting sử dụng cho việc tạo main section, nếu trong trường hợp file ko tồn tại => nó sẽ tự tạo vào thêm main section.

- ini\_subsetting sử dụng cho việc add thêm các mô tả vào file cấu hình

- yêu cầu: “puppet module install puppetlabs-inifile”

VD: Thêm main section

# Tạo file initest.pp

ini\_setting {'server\_true':

path => '/tmp/server.conf',

section => 'main',

setting => 'server',

value => 'true',

}

# chạy

$ puppet apply initest.pp

## Kết quả

$ cat /tmp/server.conf

[main]

server = true

VD: Add thêm mô tả section

ini\_subsetting {'server\_name':

path => '/tmp/server.conf',

section => 'main',

setting => 'server\_host',

subsetting => "$hostname",

}

# apply file

$ puppet apply initest.pp

# kq

[main]

server = true

server\_host = mylaptop

Using Augeas to reliably edit config files- Bình thường mỗi dạng file cấu hình đều có cấu trúc khác nhau => viết các biểu thức tìm kiếm trở nên phức tạp

- Augeas hỗ trợ đơn giản hóa việc này, đồng thời trình bày các giá trị dưới dạng cây

- Chức năng “augeas” resource

VD: type sẽ có tên là ‘enable-ip-forwarding’, file cấu hình sẽ là ‘/etc/sysctl.conf’. khi chỉ định được len, ký tự đầu tiên sẽ được viết hoa, kết thúc = ‘.lns’ .

- “changes => ['set net.ipv4.ip\_forward 1']” sẽ được hiểu ‘set <parameter> <value> => trong file config sẽ được thêm “net.ipv4.ip\_forward=1”

class base {

augeas { 'enable-ip-forwarding':

incl => '/etc/sysctl.conf',

lens => 'Sysctl.lns',

changes => ['set net.ipv4.ip\_forward 1'],

}

}

# apply

$ puppet agent -t

# kết quả

$ sysctl -p | grep ip\_forward

net.ipv4.ip\_forward = 1

Building config files using snippets- Tạo file cấu hình từ nhiều mảnh nhỏ (trong trường hợp cần nhiều class để xây dựng 1 file cấu hình)

- Sử dụng module concat, yều cầu “puppet module install puppetlabs-concat”

- trường order là sắp xếp các trạng thái sẽ được đưa vào file

VD:

# tại thuộc tính

concat {'hosts.allow':

path => '/etc/hosts.allow',

mode => 0644

}

# tạo fragment module cho header của file

concat::fragment {'hosts.allow header':

target => 'hosts.allow',

content => "# File managed by puppet\n",

order => '01'

}

# tạo concat::fragment thêm local file

concat::fragment {'hosts.allow local':

target => 'hosts.allow',

source => '/etc/hosts.allow.local',

order => '10',

}

# thêm vào cuối file

concat::fragment {'hosts.allow tftp':

target => 'hosts.allow',

content => "in.ftpd: .example.com\n",

order => '50',

}

# tạo file /etc/hosts.allow.local thêm nội dung

in.tftpd: .example.com

# apply file vào t có

$ cat /etc/hosts.allow

# File managed by puppet

in.tftpd: .example.com

in.ftpd: .example.com

Using ERB templates- templates có thể thực hiện tính toán, chạy các ruby code hoặc tham chiếu 1 số values hoặc variable từ puppet manifest

- nó sử dụng để thay thế các giá trí gán. VD: “<%= @name %>, this is a very large drink.”

- giá trị trong template sẽ tham chiếu đến giá trị được định nghĩa trong file manifest

VD: ERB template, sử dụng ERB template để insert passwd trong backup script

- Khi var tham chiếu trong template <%= @mysql\_password %>, puppet sẽ thay thế nó đến giá trị liên quan

Bước 1: tạo modules/admin/templates/backup-mysql.sh.erb

#!/bin/sh

/usr/bin/mysqldump -uroot \ -p \ --all-databases | \ /bin/gzip > /backup/mysql/all-databases.sql.gz

Bước 2: chỉnh sửa site.pp

node 'cookbook' {

$mysql\_password = 'secret'

file { '/usr/local/bin/backup-mysql':

content => template('admin/backup-mysql.sh.erb'),

mode => '0755',

}

}

Bước 3: apply file và xem kết quả

# puppet agent -t

# cat /usr/local/bin/backup-mysql

#!/bin/sh

/usr/bin/mysqldump -uroot \

-psecret \

--all-databases | \

/bin/gzip > /backup/mysql/all-databases.sql.gz

Using array iteration in templates- Truyền array vào 1 template

VD:

Bước 1: sửa file site.pp

node 'cookbook' {

$ipaddresses = ['192.168.0.1',

'158.43.128.1', '10.0.75.207' ]

file { '/tmp/addresslist.txt':

content => template('base/addresslist.erb')

}

}

Bước 2: tạo modules/base/templates/addresslist.erb

<% @ipaddresses.each do |ip| -%>

IP address <%= ip %> is present

<% end -%>

Bước 3: apply file và kq

# puppet agent -t

# cat /tmp/addresslist.txt

VD:

$interfaces = [ {name => 'eth0', ip => '192.168.0.1'},

{name => 'eth1', ip => '158.43.128.1'},

{name => 'eth2', ip => '10.0.75.207'} ]

##Template

<% @interfaces.each do |interface| -%>

Interface <%= interface['name'] %> has the address <%= interface['ip'] %>.

<% end -%>

##KQ

Interface eth0 has the address 192.168.0.1.

Interface eth1 has the address 158.43.128.1.

Interface eth2 has the address 10.0.75.207.

Using EPP templates- không phải complie thông qua Ruby, cú pháp tương tự, thay đổi dễ thấy nhất là $var thay vì @var, đuôi file là .epp

VD:

Bước 1: tạo ~/puppet/epp-test.epp

This is <%= $message %>

Bước 2: tạo file epp.pp sử dụng epp và inline\_epp

$message = "the message"

file {'/tmp/epp-test':

content => epp('/home/thomas/puppet/epp-test.epp')

}

notify {inline\_epp('Also prints <%= $message %>'):}

Bước 3: apply và kq

puppet $ puppet apply epp.pp --parser=future

puppet $ cat /tmp/epp-test

This is the message.

VD: cả epp và inline\_epp cho phép var được override lại với function call. (override lại khi truyền tham số)

file {'/tmp/epp-test':

content => epp('/home/tuphill/puppet/epp-test.epp',

{ 'message' => "override $message"} )

}

notify {inline\_epp('Also prints <%= $message %>',

{ 'message' => "inline override $message"}):}

## apply và kq

# puppet apply epp.pp --parser=future

Notice: Compiled catalog for mylaptop.pan.costco.com in environment

production in 0.85 seconds

Notice: Also prints inline override the message

Notice: Finished catalog run in 0.05 seconds

t@mylaptop ~/puppet $ cat /tmp/epp-test

This is override the message.

Using GnuPG to encrypt secrets- GnuPG tool sử dụng để encrypt các vấn đề quan trọng cần bảo mật trong source như mật khẩu

- Yêu cầu phải có gpg: cmd “puppet resource package gnupg ensure=installed”

- Chạy cmd: “puppet $ gpg --gen-key”

- Copy gpg key đã tạo tới user acc trên Puppet master:

# scp -r .gnupg puppet@puppet.example.com:

- Sử dụng (tìm hiểu thêm)

Installing packages from a third-party repository

- sử dụng khi cần install package tại repo khác, không được cung cấp mặc định

- mặc định để phải được phải add tay vào /etc/apt/sources.list.d

VD: (add 1 repo mới)

Bước 1: Tạo file: modules/admin/manifests/percona\_repo.pp

# Install Percona APT repo

class admin::percona\_repo {

exec { 'add-percona-apt-key':

unless => '/usr/bin/apt-key list | grep percona',

command => '/usr/bin/gpg --keyserver

hkp://keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A

&& /usr/bin/gpg -a --export CD2EFD2A |

apt-key add -',

notify => Exec['percona-apt-update'],

}

exec { 'percona-apt-update':

command => '/usr/bin/apt-get update',

require => [File['/etc/apt/sources.list.d/percona.list'],

File['/etc/apt/preferences.d/00percona.pref']],

refreshonly => true,

}

file { '/etc/apt/sources.list.d/percona.list':

content => 'deb http://repo.percona.com/apt wheezy main',

notify => Exec['percona-apt-update'],

}

file { '/etc/apt/preferences.d/00percona.pref':

content => "Package: \*\nPin: release o=Percona

Development Team\nPin-Priority: 1001",

notify => Exec['percona-apt-update'],

}

}

Bước 2: add vào file site.pp

node 'cookbook' {

include admin::percona\_repo

package { 'percona-server-server-5.5':

ensure => installed,

require => Class['admin::percona\_repo'],

}

}

Bước 3: apply và kq

# puppet agent -t

Comparing package versions- sử dụng cho việc so sánh các version, sử dụng Puppet's versioncmp function.

- Cú pháp: “versioncmp( A, B )”

+ 0 if A and B are equal

+ Greater than 1 if A is higher than B

+ Less than 0 if A is less than B

VD:

node 'cookbook' {

$app\_version = '1.2.2'

$min\_version = '1.2.10'

if versioncmp($app\_version, $min\_version) >= 0 {

notify { 'Version OK': }

} else {

notify { 'Upgrade needed': }

}

}

# apply và xem kết quả

# puppet agent -t

Users and Virtual Resources

Using virtual resources- giải quyết tình trạng conflix khi 2 service dùng chung các tài nguyên giống nhau. Để giải quyết nó => biến tài nguyên thành dạng resource

VD: wordpress và drupal đều sử dụng Apache, có 2 đều yêu cầu Apache package installed

|  |  |
| --- | --- |
| class wordpress {  package {'httpd':  ensure => 'installed',  }  service {'httpd':  ensure => 'running',  enable => true,  }  } | class drupal {  package {'httpd':  ensure => 'installed',  }  service {'httpd':  ensure => 'running',  enable => true,  }  } |

- Khi install nó chung trên 1 server

node 'bigbox' {

include wordpress

include drupal

}

## kết quả sẽ gây lỗi vì khai báo package httpd đến 2 lần

- => để giải quyết có thể tách package httpd thành 1 class riêng những dẫn tới phải include nhiều lần

- cách khác là sử dụng virtual resource để giải quyết

@package { 'httpd': ensure => installed }

- Khi tại resource sử dụng realize function

realize(Package['httpd'])

- Có thể khai báo bất kỳ khi nào muốn, nó sẽ ko gây ra conflix

VD: sử dụng virtual resource

class drupal {

include virtual

realize(Package['httpd'])

realize(Service['httpd'])

}

- Tạo Drupal module với content

|  |  |
| --- | --- |
| class drupal {  include virtual  realize(Package['httpd'])  realize(Service['httpd'])  } | class wordpress {  include virtual  realize(Package['httpd'])  realize(Service['httpd'])  } |

- Tại site.pp

node 'bigbox' {

include drupal

include wordpress

}

- Chạy puppet và xem kết quả

puppet agent -t

- To realize virtual resources, you can also use the collection spaceship syntax:

Package <| title = 'httpd' |>

Package <| tag = 'web' |>

Managing users with virtual resources- Khi cần phải sử dụng lại các user trong nhiều resource

node 'server' {

include user::sysadmins

}

node 'webserver' {

include user::sysadmins

include user::developers

}

- Để giải quyết, xây dựng virtual class

VD:

Bước 1: Tạo modules/user/manifests/virtual.pp

class user::virtual {

@user { 'thomas': ensure => present }

@user { 'theresa': ensure => present }

@user { 'josko': ensure => present }

@user { 'nate': ensure => present }

}

Bước 2: Tạo modules/user/manifests/developers.pp

class user::developers {

realize(User['theresa'])

realize(User['nate'])

}

Bước 3: modules/user/manifests/sysadmins.pp

class user::sysadmins {

realize(User['thomas'])

realize(User['theresa'])

realize(User['josko'])

}

Bước 4:

node 'cookbook' {

include user::virtual

include user::sysadmins

include user::developers

}

Bước 5: apply và kết quả

# puppet agent -t

VD: sử dụng tag cho việc include

@user { 'thomas': ensure => present, tag => 'sysadmin' }

@user { 'theresa': ensure => present, tag => 'sysadmin' }

@user { 'josko': ensure => present, tag => 'dev' }

User <| tag == 'sysadmin' |>

Managing users' SSH access- Quản trị các kết nối thông qua ssh keys

NOTE: release ~= incluce

VD: Tạo module ssh\_user chưa ssh\_user definition

Bước 1: tạo modules/ssh\_user/manifests/init.pp

define ssh\_user($key,$keytype) {

user { $name:

ensure => present,

}

file { "/home/${name}":

ensure => directory,

mode => '0700',

owner => $name,

require => User["$name"]

}

file { "/home/${name}/.ssh":

ensure => directory,

mode => '0700',

owner => "$name",

require => File["/home/${name}"],

}

ssh\_authorized\_key { "${name}\_key":

key => $key,

type => "$keytype",

user => $name,

require => File["/home/${name}/.ssh"],

}

}

Bước 2: tạo modules/user/manifests/virtual.pp

@ssh\_user { 'thomas':

key => 'AAAAB3NzaC1yc2E...XaWM5sX0z',

keytype => 'ssh-rsa'

}

Bước 3: Sử modules/user/manifests/sysadmins.pp

class user::sysadmins {

realize(Ssh\_user['thomas'])

}

Bước 4: sửa site.pp

node 'cookbook' {

include user::virtual

include user::sysadmins

}

Bước 5: apply và kết quả

# Puppet agent -t

Managing users' customization files- Đồng bộ các môi trường giữa các user (các file .bash\_profile or .vimrc.)

VD: tạo modules admin\_user

Bước 1: Tạo admin\_user defined type (define admin\_user) trong modules/

admin\_user/manifests/init.pp

define admin\_user ($key, $keytype, $dotfiles = false) {

$username = $name

user { $username:

ensure => present,

}

file { "/home/${username}/.ssh":

ensure => directory,

mode => '0700',

owner => $username,

group => $username,

require => File["/home/${username}"],

}

ssh\_authorized\_key { "${username}\_key":

key => $key,

type => "$keytype",

user => $username,

require => File["/home/${username}/.ssh"],

}

# dotfiles

if $dotfiles == false {

# just create the directory

file { "/home/${username}":

ensure => 'directory',

mode => '0700',

owner => $username,

group => $username,

require => User["$username"]

}

} else {

# copy in all the files in the subdirectory

file { "/home/${username}":

recurse => true,

mode => '0700',

owner => $username,

group => $username,

source => "puppet:///modules/admin\_user/${username}",

require => User["$username"]

}

}

}

Bước 2: sửa modules/user/manifests/sysadmins.pp

class user::sysadmins {

realize(Admin\_user['thomas'])

}

Bước 3: thải đổi định nghĩa modules/user/manifests/virtual.pp

@ssh\_user { 'thomas':

key => 'AAAAB3NzaC1yc2E...XaWM5sX0z',

keytype => 'ssh-rsa',

dotfiles => true

}

Bước 4: Tạo subdirectory

# mkdir -p modules/admin\_user/files/thomas

Bước 5: tạo dotfile cho user thomas

$ echo "alias vi=vim" > modules/admin\_user/files/thomas/.bashrc

$ echo "set tabstop=2" > modules/admin\_user/files/thomas/.vimrc

Bước 6: add cho note trong site.pp

node 'cookbook' {

include user::virtual

include user::sysadmins

}

Bước 7: apply và kq

puppet agent -t

Using exported resources- Tài nguyên sẽ được apply cho tất cả các node tự động

Managing Resources and Files

Using public modules

External Tools and the Puppet Ecosystem

- Có thể mở rộng Puppet với việc tạo các custom functions, resource types,

and providers

Creating custom facts- Tạo facter riêng để sử dụng.

- Facter được định nghĩa trong Ruby file.

- Add thêm facter bằng các tạo trong “lib/facter” trong module

- Để check facter sử dụng cmd: facter -p hello

- Có thể sử dụng facter để chạy code

VD:  
Bước 1: Tạo direc modules/facts/lib/facter sau đó tạo file modules/facts/lib/facter/hello.rb

Facter.add(:hello) do

setcode do

"Hello, world"

end

end

Bước 2: Sửa site.pp

node 'cookbook' {

notify { $::hello: }

}

Bước 3: chạy và kq

# puppet agent -t

##

Notice: Hello, world

Notice: /Stage[main]/Main/Node[cookbook]/Notify[Hello, world]/

message: defined 'message' as 'Hello, world'

VD:

Facter.add(:users) do

setcode do

%x{/usr/bin/who |wc -l}.chomp

end

end

# chạy về kq:

notify { "${::users} users logged in": }

Notice: 2 users logged in

- Mở rộng hơn về facter

File.open('/proc/meminfo') do |f|

f.each\_line { |line|

if (line[/^Active:/])

Facter.add(:memory\_active) do

setcode do line.split(':')[1].to\_i

end

end

end

if (line[/^Inactive:/])

Facter.add(:memory\_inactive) do

setcode do line.split(':')[1].to\_i

end

end

end

}

End

#### kq

[root@cookbook ~]# facter -p |grep memory\_

memory\_active => 63780

memory\_inactive => 58188

- Tìm hiểu thêm: <https://docs.puppet.com/facter/3.7/index.html>

Adding external facts- Tạo các facts từ file text or script

- Cần version từ 1.7 trở lên

- Cần tạo direc: “mkdir -p /etc/facter/facts.d”

VD:

Bước 1: Tạo /etc/facter/facts.d/local.txt with the following contents: model=ED-209

Bước 2: chạy và kq

[root@cookbook ~]# facter model

ED-209

Note:

+ có thể mở rộng file text ..

+ có thể sử dụng file txt, json, yaml

VD: Sử dụng script

Bước 1: Tạo /etc/facter/facts.d/users.sh

#!/bin/sh

echo users=`who |wc -l`

Bước 2: cấp quyền chạy chmod a+x /etc/facter/facts.d/users.sh

Bước 3: chạy và kq

[root@cookbook ~]# facter users

2

- Debug facter, sử dụng cmd: facter -d robin

VD:

# facter -d robin

Fact file /etc/facter/facts.d/myfacts.json was parsed but returned an

empty data set

Creating your own resource types

- Puppet cho phép tạo ra các resource, type

- type sử dụng cho việc định nghĩa resource và parameter (package type)

- provider sử dụng để imple resource type phù hợp cho các platform khác nhau (apt, yum, ..)

- Phần này đòi hỏi hiểu biết về ruby

- Custom type có thể tồn tại trong bất kỳ module, “lib/puppet/type/…”

VD: Tạo file modules/cookbook/lib/puppet/type/gitrepo.rb

Puppet::Type.newtype(:gitrepo) do => đn tên type = gitrepo

ensurable => đn trường “ensure”

newparam(:source) do => đn nơi git sẽ được lưu trữ

isnamevar => đn giá trị của source chính là $name truyền vào

end

newparam(:path) => định nghĩa path para

end

## VD:

gitrepo { 'git://github.com/puppetlabs/puppet.git':

path => '/home/ubuntu/dev/puppet',

}

DocumentationVD:

Puppet::Type.newtype(:gitrepo) do

@doc = "Manages Git repos"

ensurable

newparam(:source) do

desc "Git source URL for the repo"

isnamevar

end

newparam(:path) do

desc "Path where the repo should be created"

end

end

Validation- Sinh ra thông báo khi gán giá trị ko hợp lệ

newparam(:path) do

validate do |value|

basepath = File.dirname(value)

unless File.directory?(basepath)

raise ArgumentError , "The path %s doesn't exist" % basepath

end

end

end

Creating your own providers- Các ví dụ trên chỉ định nghĩa “gitrepo” gồm path và source. Mà ko định nghĩa cách git check repo và chạy thế nào.

- Nói đơn giản, provider cung cấp thực thi cho type

- custom provider tồn tại trong “lib/puppet/provider/TYPE\_NAME”

VD: provider cho gitrepo sẽ là git, có thể có nhiều loại provider như svn, git, cvs ..

- Ta cần install git

Bước 1: tạo modules/cookbook/lib/puppet/provider/gitrepo/git.rb

require 'fileutils'

Puppet::Type.type(:gitrepo).provide(:git) do => đăng ký provider cho gitrepo

commands :git => "git"

def create

git "clone", resource[:source], resource[:path] => tạo repo

end

def exists? => kiểm tra path có tồn tại

File.directory? resource[:path]

end

end

Bước 2:Sửa file site.pp

node 'cookbook' {

gitrepo { 'https://github.com/puppetlabs/puppetlabs-git':

ensure => present,

path => '/tmp/puppet',

}

}

Bước 3: chạy và kq

[root@cookbook ~]# puppet agent -t

Notice: /File[/var/lib/puppet/lib/puppet/type/gitrepo.rb]/ensure:

defined content as '{md5}6471793fe2b4372d40289ad4b614fe0b'

Notice: /File[/var/lib/puppet/lib/puppet/provider/gitrepo]/ensure:

created

Notice: /File[/var/lib/puppet/lib/puppet/provider/gitrepo/git.rb]/

ensure: defined content as '{md5}f860388234d3d0bdb3b3ec98bbf5115b'

- Tìm hiểu thêm tại: <https://docs.puppet.com/puppet/latest/custom_types.html>

- <https://docs.puppet.com/puppet/5.0/provider_development.html>

- <https://docs.puppet.com/puppet/latest/complete_resource_example.html>

Creating custom functions- Tạo custom functions

VD: Tạo hàm random minute

Bước 1: Tạo file modules/cookbook/lib/puppet/parser/functions/random\_ minute.rb

module Puppet::Parser::Functions => tên module -> direc -> direc

newfunction(:random\_minute, :type => :rvalue) do |args| -> Định nghĩa new func

-> tên random\_minite

-> type = giá trị

lookupvar('hostname').sum % 60 -> lấy tên host sau đó tính tổng ký tự % 60

end

end

Bước 2: sửa site.pp

node 'cookbook' {

cron { 'randomised cron job':

command => '/bin/echo Hello, world >>/tmp/hello.txt',

hour => '\*',

minute => random\_minute(),

}

}

Bước 3: chạy và kq

# puppet agent -t

- Có thể mở rộng function, nó có thể tham chiếu đến arg, truy cập các facts

VD:

# modules/cookbook/lib/puppet/parser/functions/hashtable.rb

module Puppet::Parser::Functions

newfunction(:hashtable, :type => :rvalue) do |args|

if args.length == 2

hashtable=lookupvar(args[1]).sum

else

hashtable=lookupvar('hostname').sum

end

if args.length > 0

size = args[0].to\_i

else

size = 60

end

unless size == 0

hashtable % size

else

0

end

end

end

###

node cookbook {

$hours = hashtable(24)

$minutes = hashtable()

$days = hashtable(30)

$days\_fqdn = hashtable(30,'fqdn')

$days\_ipaddress = hashtable(30,'ipaddress')

notify {"\n hours=${hours}\n minutes=${minutes}\n days=${days}\n

days\_fqdn=${days\_fqdn}\n days\_ipaddress=${days\_ipaddress}\n":}

}

### kq

Notice: hours=15

minutes=15

days=15

days\_fqdn=4

days\_ipaddress=2

Monitoring, Reporting, and Troubleshooting

- Sinh ra thông báo nhắm test, giải quyết các error

Noop – the don't change anything option- sử dụng để test hiện thi các thay đổi của puppet đối với các service, module …

- VD:

- Test file noop.pp

puppet apply noop.pp –noop

## kq khi sinh ra thay đổi

…

Notice: /Stage[main]/Main/File[/tmp/noop]/ensure: current\_value absent, should be file (noop)

..

- Mở rộng, còn có thể debug quá trình cài vào đưa log vào 1 file

- Tại agent sử dụng --debug

- Tại server sử dụng --trace

VD:

puppet master --compile cookbook.example.com --debug --trace --logdest /tmp/cookbook.log

Generating reports