**Note Puppet chú ý:**

Cmd tạo module cho code:

#puppet module generate lacoski-test

Đường dần chính chứa các environment, module, sử dụng việc quản lý source

#cd /etc/puppetlabs/code/

- environment mặc định là production, đường dẫn “#cd /etc/puppetlabs/code/environments/production/..”

Đường dẫn chứa module

#cd /etc/puppetlabs/code/modules/[tên module]

**Note thực hành:**

Code 1 manifest đơn giản:

Bước 1: Truy cập vào environment production

#cd /etc/puppetlabs/code/environments/production/manifests

Bước 2: Tạo mới 1 file .pp

#vim site.pp

….

node default { (cho tất cả các node client)

notify {"defau test machine":}

}

node master { (chỉ định cho node master)

notify {"Have a nice day!!":}

}

Bước 3: Chạy khởi tạo

#puppet agent -t

Cách tạo 1 module trong puppet:

Bước 1: Truy cập vào thư mục modules

#cd /etc/puppetlabs/code/modules

Bước 2: Sử dụng cmd để tạo modules

#puppet module generate lacoski-tranning

- Sau bước này 1 module mới đã được sinh ra với tên là tranning, do lacoski là người lập trình

Bước 3: Truy cập vào module

#cd /etc/puppetlabs/code/modules/tranning

- Cấu trúc 1 modules mới gồm có:

+ example

- init.pp

+ manifests

- init.pp

+ spec

+ classes

- …

….

- Quan tâm tới 2 direc example và manifest

Bước 3: Truy cập vào site.pp trong manifests, sửa nội dung

#vim /etc/puppetlabs/code/modules/tranning/manifests/site.pp

…..

class tranning {

notify {"This is tranning module":}

}

Bước 4: Test modules trong môi trường

- Truy cập vào directory examples

#cd /etc/puppetlabs/code/modules/tranning/example

#puppet apply init.pp

Để chạy 1 modules cho client

Bước 1: truy cập site.pp trong manifests production

#vim /etc/puppetlabs/code/environments/production/manifests/site.pp

node default { (cho tất cả các node client)

notify {"defau test machine":}

}

node master { (chỉ định cho node master)

notify {"Have a nice day!!":}

notify {"Test tranning modules":}

include tranning

}

Code:

# puppet\_service.pp

service { 'puppet':

ensure => 'stopped',

enable => false,

}

- The difference between parameters and properties is that the parameter merely indicates how Puppet should manage the resource, not what a desired state is. Puppet will only take action on property values

- test puppet conflix

# puppet apply puppet\_service.pp --noop

Struct DSL

IF

if 'mail\_lda' in $needed\_services {

service { 'dovecot': enable => true }

} else {

service { 'dovecot': enable => false }

}

CASE

case $role {

'imap\_server': {

package { 'dovecot': ensure => 'installed' }

service { 'dovecot': ensure => 'running' }

}

/\_webserver$/: {

service { [ 'apache', 'ssh' ]: ensure => 'running' }

}

default: {

service { 'ssh': ensure => running }

}

}

Selector: Là sự biến đổi của case

package { 'dovecot':

ensure => $role ? {

'imap\_server' => 'installed',

/desktop$/ => 'purged',

default => 'removed',

},

}

Xem chi tiết 1 property

#puppet describe [name]

Định nghĩa sự phụ thuộc:

- Get file from server:

package { 'haproxy':

ensure => 'installed',

before => File['/etc/haproxy/haproxy.cfg'],

}

file { '/etc/haproxy/haproxy.cfg':

ensure => file,

owner => 'root',

group => 'root',

mode => '0644',

source => 'puppet:///modules/haproxy/etc/haproxy/haproxy.cfg',

before => Service['haproxy'],

}

service { 'haproxy':

ensure => 'running',

}

Sử dụng biến:

- Biến được khai báo sau dấu $[tên]

$download\_server = 'img2.example.net'

$url = https://${download\_server}/pkg/example\_source.tar.gz

- Giá trị biến được thay thế “${}” , Đối với dấu ‘’ sẽ giữ nguyên

- Ưu điểm, làm code sáng sủa, dễ thay đổi ..

Variable types

- các loại biến hỗ trợ: Strings, Arrays, Hashes, and Boolean

$a\_bool = true

$a\_string = 'This is a string value'

$an\_array = [ 'This', 'forms', 'an', 'array' ]

$a\_hash = {

'subject' => 'Hashes',

'predicate' => 'are written',

'object' => 'like this',

'note' => 'not actual grammar!',

'also note' => [ 'nesting is',

{ 'allowed' => ' of course' } ],

}

- Gán giá trị cho biến đơn giản

$x = $a\_string

$y = $an\_array[1]

$z = $a\_hash['object']

- String thường được sử dụng như các resource attribute values, sử dụng cho việc tham chiếu các thao tác.

package { $apache\_package:

ensure => 'installed'

}

- Có truyền 1 array, sử dụng khi cần lặp lại 1 tác vụ

$packages = [ 'apache2',

'libapache2-mod-php5',

'libapache2-mod-passenger', ]

package { $packages:

ensure => 'installed'

}

Controlling the order of evaluation

- a manifest is not a script or program, nó là 1 tool sử dụng cho việc hệ thống trạng thái thông qua tập các resource, file, pack, cron ..

- Đối với puppet DSL, cac manifest sẽ định nghĩa 1 tập các resource có sẵn và sử dụng các properties được đĩnh nghĩa trước. Nói đơn giản nó tập các hành động cần thiết để nhận được 1 kết quả mong muốn

VD:

package { 'haproxy':

ensure => 'installed',

}

file {'/etc/haproxy/haproxy.cfg':

ensure => file,

owner => 'root',

group => 'root',

mode => '0644',

source => 'puppet:///modules/haproxy/etc/haproxy/haproxy.cfg',

}

service { 'haproxy':

ensure => 'running',

}

- VD: => Haproxy phải là packet được cài đặt trước => haproxy.cfg đã chó trước content được chuẩn bị trước trong /etc/puppet/modules => HA sẽ được chạy

Declaring dependencies- Để các bước trên thực hành thành 1 chuỗi là sử dụng dấu ‘->’

package { 'haproxy':

ensure => 'installed',

}

->

file { '/etc/haproxy/haproxy.cfg':

ensure => file,

owner => 'root',

group => 'root',

mode => '0644',

source => 'puppet:///modules/haproxy/etc/haproxy/haproxy.cfg',

}

->

service {'haproxy':

ensure => 'running',

}

- Cách này chỉ sử dụng được khi các bước rõ ràng, xếp đúng thứ tự, ko linh động.

- Cách khác để tạo ra các bước là sử dụng metaparameters, sử dụng tham số “require” or “before”, thường sử dụng require vì tính trực quan

package { 'haproxy':

ensure => 'installed',

}

file {'/etc/haproxy/haproxy.cfg':

ensure => file,

owner => 'root',

group => 'root',

mode => '0644',

source => 'puppet:///modules/haproxy/etc/haproxy/haproxy.cfg',

**require => Package['haproxy'],**

}

service {'haproxy':

ensure => 'running',

**require => File['/etc/haproxy/haproxy.cfg'],**

}

Hoặc

package { 'haproxy':

ensure => 'installed',

**before => File['/etc/haproxy/haproxy.cfg'],**

}

file { '/etc/haproxy/haproxy.cfg':

ensure => file,

owner => 'root',

group => 'root',

mode => '0644',

source => 'puppet:///modules/haproxy/etc/haproxy/haproxy.cfg',

**before => Service['haproxy'],**

}

service { 'haproxy':

ensure => 'running',

}

VD:

|  |  |
| --- | --- |
| file { '/etc/apache2/apache2.conf':  ensure => file,  }  file { '/etc/apache2/httpd.conf':  ensure => file,  }  service { 'apache2':  ensure => running,  enable => true,  require => [  File['/etc/apache2/apache2.conf'],  File['/etc/apache2/httpd.conf'],  ],  } | file { '/etc/apache2/apache2.conf':  ensure => file,  before => Service['apache2'],  }  file { '/etc/apache2/httpd.conf':  ensure => file,  before => Service['apache2'],  }  service { 'apache2':  ensure => running,  enable => true,  } |

VD: Restart service sau khi đè file cấu hình

|  |
| --- |
| if $os\_family == 'Debian' {  file { '/etc/apt/preferences.d/example.net.prefs':  content => '…',  before => Package['apache2'],  }  }  package { 'apache2':  ensure => 'installed',  } |

Thông báo lỗi

- Nó sẽ được hiện thị dưới dạng các “Notice” hoặc “Warning”

Tránh sự phụ thuộc vòng (đệ quy các bước)

- Khi sinh ra lỗi liên quan sử dụng cmd để hiện thị ra các mối quan hệ

puppet apply circle.pp --graph

Note:

- If a directory and a file inside the directory is declared, Puppet will first create the directory and then the file

- If a user and his primary group is declared, Puppet will first create the group and then the user

- If a file and the owner (user) is declared, Puppet will first create the user and then the file

Implementing resource interaction

Examining the most notable resource types

- Các resource thường dùng

File

|  |  |
| --- | --- |
| # tạo 1 file + content  file { '/etc/modules':  ensure => file,  content => "# Managed by Puppet!\n\ndrbd\n",  } | # tạo sym link  file { '/etc/apache2/sites-enabled/001-puppet-lore.org':  ensure => 'link',  target => '../sites-available/puppet-lore.org',  } |

Package

|  |
| --- |
| # chỉ định packet, cài nếu ko có  package { 'haproxy':  ensure => present,  provider => 'dpkg',  source => '/opt/packages/haproxy-1.5.1\_amd64.dpkg',  } |

Service

|  |
| --- |
| # kiếm tra, chạy service, nếu file cấu hình thay đổi, service sẽ được restart (subcribe)  service { 'count-logins':  provider => 'base',  ensure => 'running',  binary => '/usr/local/bin/cnt-logins',  start => '/usr/local/bin/cnt-logins --daemonize',  subscribe => File['/usr/local/bin/cnt-logins'],  } |

The user and group types- Tạo group-user, quản trị user group dễ dàng với puppet

- Sẽ báo warning nếu attri ko chính xác, hoặc không hỗ trợ

|  |
| --- |
| group { 'proxy-admins':  ensure => present,  gid => 4002,  }  user { 'john':  ensure => present,  uid => 2014,  home => '/home/john'  managehome => true, # <- adds -m to useradd  gid => 1000,  shell => '/bin/zsh',  groups => [ 'proxy-admins' ],  } |

The exec resource type

- Puppet không phải là 1 script, thay vì đó, nó cung cấp các tool cho phép tương tác với hệ thống thông qua DSL, đồng bộ với các nền tảng khác nhau

- exec cho phép thực hiện 1 số cmd cho hệ điều hành (phải được hỗ trợ câu lệnh)

VD:

|  |
| --- |
| # giải nén gói  exec { 'tar cjf /opt/packages/homebrewn-3.2.tar.bz2':  cwd => '/opt',  path => '/bin:/usr/bin',  creates => '/opt/homebrewn-3.2',  } |

The cron resource type

- Các thiệt lập cho cmd và cron

|  |
| --- |
| cron { 'clean-files':  ensure => present,  user => 'root',  command => '/usr/local/bin/clean-files',  minute => '1',  hour => '3',  weekday => [ '2', '6' ],  environment => 'MAILTO=felix@example.net',  } |

The mount resource type- Tự động mount giá trị

mount { '/media/gluster-data':

ensure => 'mounted',

device => 'gluster01:/data',

fstype => 'glusterfs',

options => 'defaults,\_netdev',

dump => 0,

pass => 0,

}

The Master and Its Agents

The Puppet MasterCác nhiệm vụ của master:

- Storing and compiling manifests

- Serving as the SSL certification authority

- Processing reports from the agent machines

- Gathering and storing information about the agents

Puppet Master and Puppet Server

- Puppet's master cung cấp RESTful HTTP API

- Agents giao tiếp thông qua HTTPS transactions, được chứng thực ssl

Setting up the master machine

- Port Puppet master: 8140

Creating the master manifest- Để định nghĩa các manifest thuộc về agent, viết nó trong các node

VD:  
**node 'agent' {**

$packages = [ 'apache2',

'libapache2-mod-php5',

'libapache2-mod-passenger', ]

package { $packages:

ensure => 'installed',

before => Service['apache2'],

}

service { 'apache2':

ensure => 'running',

enable => true,

}

}

Inspecting the configuration settings- All the customization of the master's parameters can be made in the puppet.conf file.

- cmd lấy đường dẫn mặc định manifest

#puppet master --configprint manifest

/etc/puppetlabs/code/environments/production/manifests

Setting up the Puppet Agent

- sử dụng gói puppet-agent

- sau khi thiết lập xong các config cơ bản, sử dụng

#puppet agent –test

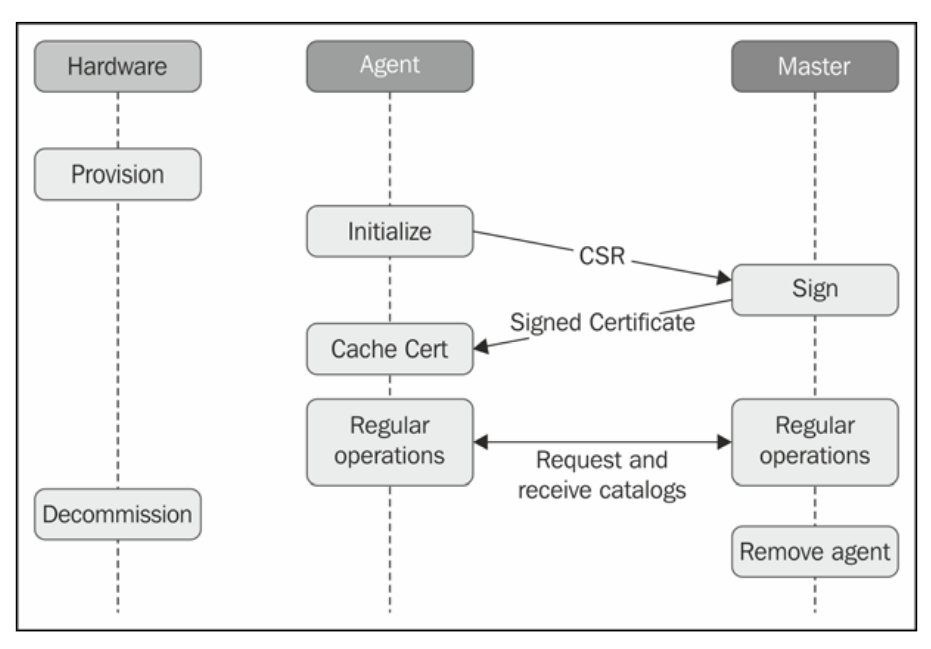
- kết nối agent master theo tài liệu triển khai

The agent's life cycle- Cách bước kết nối agent đến master

Bước 1: generating a key and a certificate signing request, xin cấp ssl

Bước 2: server nhận kết nối, cấp chứng chỉ

Bước 3: client nhận được chứng chỉ, hoàn thành kết nối



- Sau 1 thời gian nhất định, agent sẽ check update catalogs, mặc đinh 30p

Xóa chứng thực

- Cmd xóa chứng chỉ hiện tại của các client:

# puppet cert revoke agent

- Sau cmd trên các client sẽ phải kết nối lại với master

# puppet agent –test

- có thể có lỗi do master vẫn lưu cache client, để sửa

Trên master:

# puppet cert clean agent.example.net

Trên agent:

# find /etc/puppetlabs/puppet/ssl -name agent.example.net.pem –delete

# puppet agent –t

Performance considerations- master cung cấp share file, cung cấp file source đến các agent

file { '/usr/local/etc/my\_app.ini':

ensure => file,

owner => 'root',

group => 'root',

source => 'puppet:///modules/my\_app/usr/local/etc/my\_app.ini',

}

Using Phusion Passenger with NginxCompleting the stack with PuppetDBTroubleshooting SSL issues

A Peek under the Hood – Facts, Types, and Providers

Summarizing systems with Facter- Mỗi môi trường có 1 biến giá trị giống nhau để phản ảnh 1 trạng thái, giá trị của hệ thống, có thể cùng tên hoặc khác tên => puppet cung cấp các biến toàn cục, đồng bộ đó, dễ dàng cho việc so sánh, sử dụng => nó thuộc tập facter hay các facts

VD

|  |  |
| --- | --- |
| if `grep -c ^processor /proc/cpuinfo` > 2 {  $load\_warning = 4  }  else {  $load\_warning = 2  } | if $::processorcount > 4 { … } |

VD:

puppet apply -e 'notify { "I am ${::fqdn} and have ${::processorcount} CPUs": }'

- Cấu trúc của facts, các biến này ko thể tạo hoặc gán

+ $facts[::operatingsystem]

+ $::operatingsystem

VD:

file { '/etc/my-secret':

ensure => 'file',

mode => '0600',

owner => 'root',

source => "puppet:///modules/

secrets/${::clientcert}/key",

}

VD:

if $::operatingsystem != 'Ubuntu' {

package { 'avahi-daemon':

ensure => absent

}

}

VD:

if $::operatingsystem == 'Debian' {

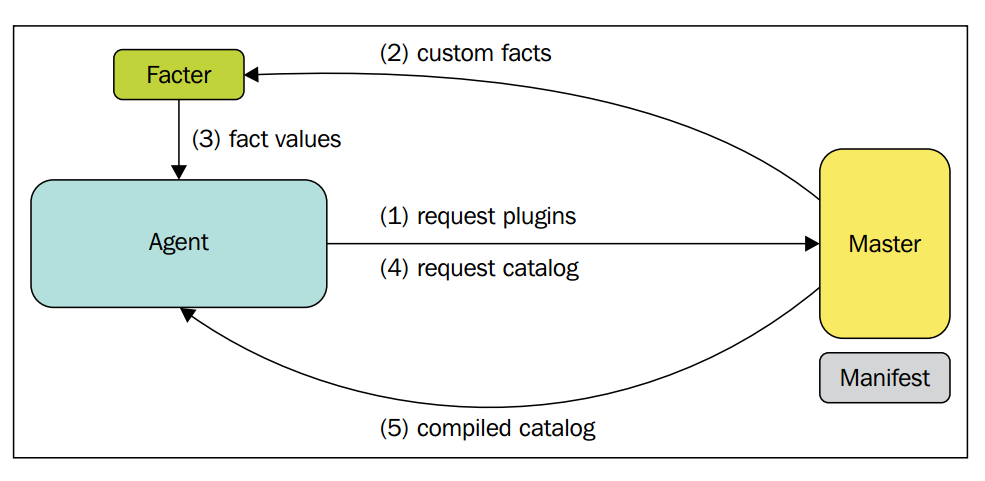
if versioncmp($::operatingsystemrelease, '7.0') >= 0 {

$ssh\_ecdsa\_support = true

}

}

Extending Facter with custom facts



Goals of Facter

- Sử dụng cho việc phát triển hệ thống đa nền tảng, dựa trên tập các fasts có sẵn

- Facter tạo ra sự trừ tượng, giúp việc phát triển hệ thống trở nên dễ dàng hơn

Understanding the type system

- Các resource đại diện các trạng thái của agent, các resource type, name, list atrribute.

- attribute có thể là property hoặc parameter

- properties represent distinct pieces of state

- parameters merely influence Puppet's actions upon the property values.

Cơ chế hoạt động Puppet

- agent thực hiện tất cả công việc trong qua transaction. Các transaction hoạt động khi

+ tiến trình background agent được kích hoạt

+ command --onetime or --test

The transaction always passes several stages. They are as follows: 1. Gathering fact values to form the actual catalog request.

2. Receiving the compiled catalog from the master.

3. Prefetching of current resource states.

4. Validation of the catalog's content.

5. Synchronization of the system with the property values from the catalog

The resource type's life cycle on the agent side

- khi master đã compile catalog, cata sẽ được agent validation

Substantiating the model with providers

Modularizing Manifests with Classes and Defined Types

Introducing classes and defined types- có 2 loại classes và defined types

- function có chút tương tự với class nhưng nó phù hợp cho việc định nghĩa type

- chúng nhìn có vẻ giống nhau nhưng khác cách sử dụng

Defining and declaring classes- Puppet class có thể coi là nơi chứa các resource

- Được định nghĩa 1 lần và được lựa chọn sử dụng cho các node

- mỗi class đại diện cho 1 tập các cấu hình hệ thống

VD:

|  |
| --- |
| class apache {  package { 'apache2':  ensure => 'installed',  }  file { '/etc/apache2/apache2.conf':  ensure => 'file',  source => 'puppet:///modules/apache/etc/apache2/apache2.conf',  require => Package['apache2'],  }  service { 'apache2':  enable => true,  subscribe => File['/etc/apache2/apache2.conf',  }  } |

- Tại manifest, node block add thêm “include apache”

|  |
| --- |
| node 'webserver01' {  include apache  } |

Creating and using defined types- Giống như việc lập ra 1 kịch bản cho manifest

- Nó có thể nhập đối số, linh động các giá trị

VD:

|  |
| --- |
| define virtual\_host(  String $content,  String[3,3] $priority = '050'  ) {  file { "/etc/apache2/sites-available/${name}":  ensure => 'file',  owner => 'root',  group => 'root',  mode => '0644',  content => $content  }  file { "/etc/apache2/sites-enabled/${priority}-${name}":  ensure => 'link',  target => "../sites-available/${name}";  }  } |

- Tại manifest

|  |
| --- |
| virtual\_host { 'example.net':  content => file('apache/vhosts/example.net')  }  virtual\_host{ 'fallback':  priority => '999',  content => file('apache/vhosts/fallback')  } |

VD: trong trường hợp nhiều type

virtual\_host {

'example.net':

content => 'foo';

'fallback':

priority => '999',

content => ...,

}

Understanding and leveraging the differences- class khai báo các resource, properties thuộc chính giữa hệ thống, mô tả về 1 phần hoặc toàn bộ hệ thống. class giống form, duy nhất, các thông tin được fix cứng

- các resource được đóng gói vào class dễ dàng cho việc khai báo:

+ One or more packages that should be installed (or removed)+ A specific configuration file in /etc

+ A common directory, needed to store scripts or configs for many subsystems

+ Cron jobs that should be mostly identical on all applicable systems

- Define được sử dụng cho rất nhiều vấn đề, tùy thuộc vào các trường hợp, vd:

+ Files in a conf.d style directory

+ Entries in an easily parseable file such as /etc/hosts

+ Apache virtual hosts

+ Schemas in a database

+ Rules in a firewall

Structured design patterns

Writing comprehensive classes (bao hàm class)

- 1 tập các class nhằm mục đích cấu hình 1 dịch vụ, vd apache

- các vấn đề trong apache:

+ The firewalling software is installed and configured with a default ruleset+ The malware detection software is installed

+ Cron jobs run the scanners in set intervals

+ The mailing subsystem is configured to make sure the cron jobs can deliver

their output

- Có 2 cách để tạo thiết kế class cho 1 vấn đề lớn:

+ thiết kế 1 class lớn với đầy đủ các resource, thiết lập đầy đủ cho 1 dịch vụ, hệ thống

+ tạo các class nhỏ, mỗi class có 1 vài resource, cùng với include statements các class đơn giản. các chức năng được chia nhỏ và có 1 class tập hợp

VD:

|  |
| --- |
| class monolithic\_security {  package { [ 'iptables', 'rkhunter', 'postfix' ]:  ensure => 'installed';  }  cron { 'run-rkhunter':  ...  }  file { '/etc/init.d/iptables-firewall':  source => ...  mode => 755  }  file { '/etc/postfix/main.cf':  ensure => 'file',  content => ...  }  service { [ 'postfix', 'iptables-firewall' ]:  ensure => 'running',  enable => true  }  }  class divided\_security {  include iptables\_firewall  include rkhunter  include postfix  } |

- Khi thiết kế các class, ko nên cố gắng phân tách thành các thái cực, thiết kế class theo sở thích và hợp lý:

+ tập chung vào các class nguyên khối

+ tránh việc chia class => dẫn đến conflix data

- case-by-case design choices will be based on each author's experience and preference

Writing component classes- chia mỗi class nhằm giải quyết 1 số mục đích nhất định, giới hạn chắc năng class

- Ví dụ, đối với các packet sẽ được sử dụng lặp nhiều lần trong 1 hệ thống => tách riêng ra 1 class

- cách tốt nhất để chia class là nắm rõ các chức năng, mục tiêu của hệ thống cần triển khai

Using defined types as resource wrappers

- definded type có cách sử dụng khác class, thường nhắm vào các chức năng giống nhau, chỉ khác các tham số truyền vào. Vd: ghi 1 nội dung vào 1 file, lấy 1 file từ server

|  |  |
| --- | --- |
| module\_file { '/etc/ntpd.conf':  module => 'ntp':  } | define module\_file(String $module) {  file { $name:  source => "puppet:///modules/${module}/${name}"  }  }  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  define module\_file(  String $module,  Optional[String] $mode = undef  ) {  if $mode != undef {  File { mode => $mode }  }  file { $name:  source => "puppet:///modules/${module}/${name}"  }  } |

Mở rộng defined type

- khi muốn mở rộng chức năng của type

VD:

|  |
| --- |
| String $key,  Optional[String] $uid = undef,  String $group = 'users'  ) {  user { $title:  ensure => present  gid => $group,  managehome => true,  } ->  ssh\_authorized\_key { "key for ${title}":  ensure => present,  user => $title,  type => 'rsa',  key => $key,  }  } |

Using defined types as macros

- dùng cho sự lặp lại

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | file { '/etc/example\_app/conf.d.enabled/england':  ensure => 'link',  target => '../conf.d.available/england'  }  file { '/etc/example\_app/conf.d.enabled/ireland':  ensure => 'link',  target => '../conf.d.available/ireland'  }  file { '/etc/example\_app/conf.d.enabled/germany':  ensure => 'link',  target => '../conf.d.available/germany'  ...  } | define example\_app\_config {  file { "/etc/example\_app/conf.d.enabled/${name}":  ensure => 'link',  target => "../conf.d.available/${name}",  }  }  Run:  example\_app\_config {'england': }  example\_app\_config {'ireland': }  example\_app\_config {'germany': }  or  example\_app\_config { [ 'england', 'ireland', 'germany', ... ]:  } | |

Exploiting array values using defined types(Sử dụng array cho việc defiend type)

- Từ ví dụ, ta có thể tối ưu làm sao chỉ cần truyền 1 array vào defined type, việc sử lý sữ diện ra bên trong các type được defined

VD: 1 đất nước có nhiều thành phố => ko thể lặp quá trình này thủ công, hoặc làm defined type trở nên phức tạp => tách riêng thành 2 type

|  |  |
| --- | --- |
| define example\_app\_config {  file { "/etc/example\_app/conf.d.enabled/${name}":  ensure => 'link',  target => "../conf.d.available/${name}",  }  }  example\_app\_config {'england': }  example\_app\_config {'ireland': }  example\_app\_config {'germany': }  or  example\_app\_config { [ 'england', 'ireland', 'germany', ... ]:  } | define example\_app\_config (  Array $regions = []  ) {  file { "/etc/example\_app/conf.d.enabled/${name}":  ensure => link,  target => "../conf.d.available/${name}",  }  example\_app\_region { $regions:  country => $name,  }  }  define example\_app\_region(String $country) {  file { "/etc/example\_app/conf.d.enabled/${country}/regions/${name}":  ensure => 'link',  target => "../../regions.available/${name}",  }  }  example\_app\_config { 'england':  regions => [ 'South East', 'London' ],  }  example\_app\_config { 'ireland':  regions => [ 'Connacht', 'Ulster' ],  }  example\_app\_config { 'germany':  regions => [ 'Berlin', 'Bayern', 'Hamburg' ],  } |

Using iterator functions(sử dụng các hàm lặp)

- Tính năng mới của puppet từ >= 4, cho phép định nghĩa các vòng lặp nhanh chóng, sd each và map functions

VD: Định nghĩa nhanh các country

|  |
| --- |
| [ 'england', 'ireland', 'germany' ].each |$country| {  file { "/etc/example\_app/conf.d.enabled/${country}":  ensure => 'link',  target => "../conf.d.available/${country}",  }  } |

VD: Đối với các kiểu phức tạp

$region\_data = {

'england' => [ 'South East', 'London' ],

'ireland' => [ 'Connacht', 'Ulster' ],

'germany' => [ 'Berlin', 'Bayern', 'Hamburg' ],

}

$region\_data.each |$country, $region\_array| {

$region\_array.each |$region| {

file { "/etc/example\_app/conf.d.enabled/${country}/

regions/${region}":

ensure => link,

target => "../../regions.available/${region}",

}

}

}

Including classes from defined types- Kết hợp defined type và class,

|  |  |
| --- | --- |
| file { [ '/etc/example\_app', '/etc/example\_app/config.d.enabled' ]:  ensure => 'directory',  } | define example\_app\_config(Array $regions = []) {  include example\_app\_config\_directories  ...  } |

Establishing relationships among containers

- class trong puppet ko tương đồng với class trong opp ngôn ngữ lập trình, nó ko có methods, ko có attribute, ko thể tạo interface, abstract

Passing events between classes and defined types

- collection of classes and defined types as your interface

- Sự bất tiện xuất hiện khi đống gói tất cả class và type => việc chỉnh sửa 1 số cài đặt hệ thống trở nên thiếu linh hoạt

Vd:

|  |  |
| --- | --- |
| file { '/var/lib/apache2/sample-module/data01.bin':  source => '...',  notify => Class['apache'],  }  service { 'apache-logwatch':  enable => true,  subscribe => Class['apache'],  } | $active\_countries = [ 'England', 'Ireland', 'Germany' ]  service { 'example-app':  enable => true,  subscribe => Symlink[$active\_countries],  }  $active\_countries = [ 'England', 'Ireland', 'Germany' ]  service { 'example-app':  enable => true,  subscribe => Symlink[$active\_countries],  } |

Ordering containers- Sử dụng các metaparameter để điều hướng các tùy chỉnh: notify, subscribe, before, before

VD: tất cả resource của firewall sẽ được load trước khi thực thị loadbalancing

include firewall

include loadbalancing

Class['firewall'] -> Class['loadbalancing']

Limitations- một số vấn đề xuất hiện khi sử dụng các sự phụ thuộc như sẽ load lại toàn bộ các resource của 1 type hoặc 1 class, restart nó trong khi công việc chỉ cần thay đổi 1 phần nhỏ, không cần gây ảnh hướng đến toàn hệ thống, nhất là các class liên quan đến các class khác

- có thể giới hạn các phần sẽ được sử dụng trong các depen

NOTE:

- Cần cân nhắc khi đặt mỗi inclune, nếu ko chỉ cần 1 thay đổi nhỏ sẽ dẫn đến hàng loạt các thay đổi lớn

VD:

|  |  |
| --- | --- |
| file { '/etc/apache2/conf.d/passwords.conf':  source => '...',  require => Class['apache'],  notify => Class['apache'],  } | file { '/etc/apache2/conf.d/passwords.conf':  source => '...',  require => Class['apache::package'],  notify => Class['apache::service'],  } |

The performance implications of container relationships (các vấn đề về hiệu suất)

- Puppet agent khi build các source sẽ xây dựng lên 1 graph về các sự phụ thuộc, sự phụ thuộc càng lớn thì puppet agent sẽ càng mất thời gian xử lý và debug

=> cần lưu ý về các depend trong khi xây dựng các gói

Mitigating the limitations