**Tìm hiểu environments**

**Khái niệm Environments**

- Environments sử dụng cho việc cô lập các nhóm Puppet agent

- Puppet master cung cấp mỗi môi trường các main manifest và module riêng, sử dụng các version khác nhau của cùng 1 module khác group, dẽ dàng cho việc test

Access environment name in Puppet manifests

- sử dụng $environment để lất tên của môi trường hiện tại

**Environments scenarios**

3 lý do chính để sử dụng Environment:

- test môi trường tạm thời

- test môi trường dài hạn

- phân chia cơ sở hạ tầng

**Permanent test environments**

- Trong môi trường này, các group test node đã ổn định khi cài đặt các phiên bản mới, trước khi merged với source code chính.

**Temporary test environments**

- Sử dụng cho việc test các thay đổi nhỏ, hoặc test vấn đề triển khai 1 số dịch vụ lặp đi lắp lại

**Divided infrastructure**

- Nếu 1 hạ tầng được quản lý bởi nhiều team khác nhau

Mốt số giới hạn môi trường

<https://docs.puppet.com/puppet/5.0/environments_about.html>

**Creating Environments**

**Environment structure**

- Mỗi môi trường được coi là 1 nhánh, được thể hiện trong thưc mục Puppet master

Một số lưu ý khi tạo environment:

- nó sẽ bao gồm modules directory

- nó sẽ bao gồm manifests directory

- nó có bao gồm hiera.yaml (bản 5)

- có thể bao gồm environment.conf (override 1 số setting) bao gồm modulepath, manifest

**Environment resources**

- Environment chỉ ra resource mà puppet master sẽ compile catalog có các agent nodes.

- Các modulepath, the main manifest, hiera data, and the config version script sẽ được chỉ định trong environment.conf

**The modulepath**

- tập các direc Puppet sẽ load module

- Mặc định, puppet sẽ load module từ environment’s modules directory, sau đó đến puppet.conf basemodulepath.

- Nếu module trống, puppet sẽ load từ basemodulepath

**The main manifest**

- Điểm puppet compile catalogs

- trừ khi override environments, puppet sẽ mặc định sử dụng global default\_manifest cho tùy chỉnh

- The value of this setting can be an absolute path to a manifest that all environments will share, or a relative path to a file or directory inside each environment.

- giá trị mặc định default\_manifest là ./manifests - environment’s own manifests directory

- Nếu giá trị default\_manifest trống, nó sẽ tạo file trắng. có thể chỉ định thông qua puppet.conf

**Hiera data**

**The config version script**

- Puppet sẽ tự sinh ra config version cho mỗi compile catalogs như các message cho việc report. Version là 1 phần của data sử dụng cho việc xác định catalog, events

The environment.conf file

- environment có thể chứa file environment.conf, sử dụng để override các giá trị tùy chọn

- environment.conf sẽ override: modulepath, manifest, config\_version, environment\_timeout

**Create an environment**

Bước 1:

- Tại code directory tạo thư mục enviroments

Bước 2:

- Trong environments directory, tạo directory với tên của environment mới cần tạo, sử dụng cấu trúc $codedir/environments/

Bước 3:

- Tạo modules direc và manifests direc

**Additional steps**

**Cấu hình modulepath**

Bước 1:

- set modulepath trong environment.conf, nếu muốn tạo dạng globle thì sử dụng modulepath trong puppet.conf

Bước 2:

- Kiểm tra modulepath với môi trường chỉ định = cmd:

puppet config print modulepath --section master --environment test /etc/puppetlabs/code/environments/test/modules:/etc/puppetlabs/code/modules:/opt/puppetlabs/puppet/modules

**Configure a main manifest:**

Bước 1:

* Set manifest trong environment.conf. Set toàn cục trong default\_manifest, có thể sử dụng đường dẫn tương đối hoặc tuyệt đối

Bước 2:

* Khóa tất cả environments tới single global manifest với trường disable\_per\_environment\_manifest, ngăn chặn bất kỳ environment setting thuộc own manifest

**Để chỉ định executable script xác định environment’s config :**

Bước 1:

* Set thuộc tính config\_version trong environment.conf. Puppet sẽ chạy scripts khi compiling catalog cho node trong environment, sử dụng đầu ra của nó cho config version

**Assign nodes to environments via an ENC**

* Có thể gán agent node tới environments bằng cách sử dụng external node classifier (ENC). Mặc định tất cả các node được gán vào environment tên production

**Assign nodes to environments via the agent’s config file**

* Có thể gán agent node tới environments bằng cách sử dụng agent’s config file. Mặc định tất cả các node được gán vào environment tên production
* Cấu hình agents sử dụng environment = cách edit puppet.conf

Bước 1:

* Mở file puppet.conf = file editor

Bước 2:

* Tìm environment setting nằm 1 trong 2 agent hoặc main section

Bước 3:

* Set value của environment setting = tên của name environment bạn muốn gán cho agent

**Global settings for configuring environments**

Setting trong master puppet.conf, đây là cách Puppet tìm và sử dụng các biến mối trường

**Environmentpath**

- Là tập các biến môi trường, nơi Puppet tìm kiếm environments, biến mặc định của nó là $codedir/environments

- Nếu có nhiều direc, sử dụng dấu “:” để tách các đường dẫn. VD: $codedir/temp\_environments:$codedir/environments

- Nếu 2 biến giá trị môi trường giống nhau, nó sẽ sử dụng thằng đâu tiên

- environmentpath sẽ nằm trong main section

**Basemodulepath**

- basemodulepath là tập các global module mà tất cả các environment có thể truy cập

- 1 số modules có thể cài mặc định cho tất cả environments

- basemodulepath được cấu hình mặc định global module directory. Mặc định $codedir/modules cho user-access module và /opt/puppetlabs/puppet/modules cho system modules

default\_manifest

- default\_manifest chỉ ra main manifest cho tất cả các môi trường mà ko phải set trong environment.conf

- giá trị mặc định default\_manifest là ./manifests

disable\_per\_environment\_manifest

environment\_timeout

[https://docs.puppet.com/puppet/5.0/environments\_creating.html#the-config-version-script](https://docs.puppet.com/puppet/5.0/environments_creating.html" \l "the-config-version-script)

**Environment isolation (Cô lập môi trường)**

- Cô lập các môi trường nhắm tránh gẫy ra xung đột giữa các mỗi tường

**Enable environment isolation with Puppet**

- Cô lập môi trường = cách sinh ra file metadata mà puppet sẽ sử dụng thay thế cho giá trị default Ruby resource

Bước 1:

- Tại cmd, chạy cấu lệnh “puppet generate types --environment <ENV\_NAME>” cho mỗi mồi trường

puppet generate types --environment production

Bước 2:

- Khi triển khai 1 phiên bản mới của Puppet, ghi đè lại metadata = cách sử dụng câu lệnh “puppet generate types --environment <ENV\_NAME> --force”

**Troubleshooting environment isolation**

- Nếu các cmd ko thể sinh ra môi trường, sinh ra lỗi thì sẽ nhận được thông báo error từ catalog compilation: “type not found” or “attribute not found.”

**Để sửa lỗi này**

Bước 1:

- Chắc chắn thuộc tính Puppet resource chinh xác.

Bước 2:

- Sinh ra lại metadata bằng cách xóa .resource\_types directory, chạy lại generate types cmd

Bước 3:

- Nếu tiếp tục nhận được catalog compilation errors, disable environment isolation

**Để disable environment isolation trong Puppet Enterprise:**

Bước 1:

- Tại /etc/puppetlabs/puppetserver/conf.d/pe-puppet-server.conf, xóa pre-commit-hook-commands setting

Bước 2:

- Tại Hiera, set puppet\_enterprise::master::puppetserver::pre\_commit\_hook\_commands: []

Bước 3:

- Tại cmd, chạy “service pe-puppetserver reload”

Bước 4:

- Xóa .resource\_types directories tại /etc/puppetlabs/code-staging.

Bước 5:

- Triển khai môi trường

**generate types**

- Khi sử dụng “generate types” cmd, nó sẽ quét tất cả các tài nguyên có sẵn trong environments, trử core Puppet resource types.

“generate types” cmd cho phép các option:

- --environment <ENV\_NAME>: sinh ra môi trường meta data, nếu ko chỉ định tham số cụ thế, nó sẽ sinh metadate cho default environment (production)

- --force: sử dụng cho việc override các metadata trước

Trên mỗi resource type implementation nó tìm thấy, command sẽ sinh ra metadate liên kết tới, đặt tên sau resource type, trong <env-root>/.resource\_types direc, nó cũng sẽ đồng bộ, vì thế:

- Types that have been removed in modules are removed from resource\_types.

- Types that have been added are added to resource\_types.

- Types that have not changed (based on timestamp) are kept as is.

- Types that have changed (based on timestamp) are overwritten with freshly generated metadata.