**SỬ DỤNG JINJA2 TRONG PYTHON ĐỂ TẠO RA FILE CẤU HÌNH**

Hãy cùng xem dữ liệu được ghi như bên dưới.

SW1:

vlans:

- id: 10

name: data

- id: 20

name: voice

- id: 30

name: DMZ

interfaces:

- id: Fa0/1

vlan: 10

mode: access

- id: Fa0/2

vlan: 20

mode: access

- id: Fa0/3

vlan: 30

mode: access

- id: Fa0/23

mode: trunk

vlan\_list: 10,20,30

- id: Fa0/24

mode: trunk

vlan\_list: 10,20,30

Bên trên là một dạng dữ liệu mô tả bằng định dạng YAML mô tả các thông tin cần phải cấu hình trên một thiết bị switch. Tại sao lại mô tả bằng định dạng YAML? Bởi vì YAML dễ viết và dễ đọc. Một kỹ sư mạng không cần có kiến thức lập trình khi đọc dữ liệu trên cũng có thể hiểu nôm na là:

* Thiết bị đang được đề cập là SW1
* Trên thiết bị có nhiều vlan, trong đó vlan 10 có tên là data, vlan 20 có tên là voice và vlan 30 có thên là DMZ
* Trên thiết bị có nhiều cổng kết nối ở những vlan khác nhau, trong đó cổng Fa0/1 là cổng access thuộc vlan 10, cổng Fa0/2 là cổng access thuộc vlan 20, cổng Fa0/3 là cổng access thuộc vlan 30; còn cổng Fa0/23 và Fa0/24 là cổng trunk, trên cổng trunk cho phép các vlan 10, 20 và 30

Bên trên là góc nhìn của một người không có kiến thức về lập trình. Còn nếu có kiến thức lập trình Python cơ bản và hiểu biết về các kiểu dữ liệu của Python thì sẽ nhìn dữ liệu trên với góc độ như sau:

* Nhìn tổng quan thì đây là dữ liệu mô tả bằng một dictionary trong đó liệt kê bằng các cặp khoá và giá trị (key-value pair)
* Khóa (Key) bên ngoài cùng là ‘SW1’ có 2 giá trị (value) là 2 danh sách (list) vlans và interfaces
* Nhìn xa hơn là nếu biết khoá (key) SW1 thì có thể lấy được dữ liệu (value) của nó. Ví dụ biết SW1 thì có thể lấy được giá trị vlans và interfaces. Lấy được lần lượt key SW1 rồi đến key vlans, sẽ thấy danh sách các vlan 10, 20, 30 cùng với tên tương ứng của những vlan đó
* Nhìn kỹ hơn, các danh sách (list) vlans và interfaces này lại chứa các dictionary trong nó. Ví dụ vlans chứ các python dictionary lưu thông tin về vlan id, vlan name.

(dấu hiệu nhận biết một dictionary là dấu : và dấu hiệu nhận biết một list là dấu - )

Đọc file YAML này và sử dụng như thế nào trong Python? Câu trả lời là dùng module yaml trong python mà người ta đã viết sẵn. Module này chuyển xử lý dữ liệu mô tả bằng định dạng yaml thành một python dictionary giúp cho việc bóc tách dữ liệu một cách dễ dàng.

Thử mở file và kiểm tra kiểu dữ liệu của thông tin trả về. ‘vlan.yaml’ là tên file được lưu trên máy tính và cùng thư mục với python script bên dưới.

import yaml

with open('vlan.yaml', 'r') as f:

data = yaml.safe\_load(f.read())

print(type(data))

Chạy thử script và in ra kiểu dữ liệu, kết quả đúng là kiểu dữ liệu dictionary. vlan\_creation.py là file python script lưu đoạn code trên.

A screen shot of a computer

Description automatically generated

Thử in ra dữ liệu đọc được ra, dùng module pprint thay vì print thông thường để dữ liệu in ra cho đẹp mắt.

from pprint import pprint

import yaml

with open('vlan.yaml', 'r') as f:

data = yaml.safe\_load(f.read())

pprint(data,indent=2)

Dữ liệu in ra đúng là theo dạng python dictionary, bên trong lại là các python list, bên trong python list lại là python dictionary.

A screen shot of a computer

Description automatically generated

Thử bóc xem giá trị của SW1 như thế nào. Truy cập giá trị của key SW1 bằng cách data[‘SW1’]

import yaml

from pprint import pprint

with open('vlan.yaml', 'r') as f:

data = yaml.safe\_load(f.read())

pprint(data['SW1'])

Giá trị của SW1 là 2 python list, trong mỗi python list đó lại là các dictionary

A screen shot of a computer

Description automatically generated

Thử duyệt danh sách các interfaces, nếu interface nào ở chế độ access thì in ra màn hình. Vì đây là một list trong python, nên để đọc lần lượt các phần tử trong nó thì cần dùng vòng lặp for.

import yaml

from pprint import pprint

with open('vlan.yaml', 'r') as f:

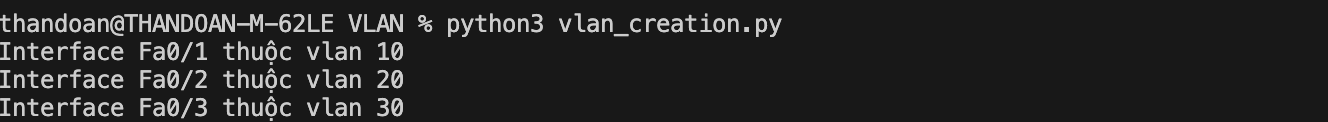
data = yaml.safe\_load(f.read())

for interface in data['SW1']['interfaces']:

if interface['mode']=='access':

print(f"Interface {interface['id']} thuộc vlan {interface['vlan']}")

Đoạn code trên duyện danh sách các interface trên SW1 (danh sách này là dictionary trong dictionary nên cần phải gọi 2 lần key: key thứ nhất là SW1, key thứ 2 là interfaces. Gọi đến đây thì ta lấy được một python list). Nếu mode của interface là access thì in ra màn hình là interface đó thuộc vlan nào.



Về cơ bản đến đây chúng ta đã biết cách đọc một file dữ liệu mô tả bằng định dạng yaml, chuyển định dạng sang dictionary trong python và bóc tách dữ liệu trong dictionary đó.

**ỨNG DỤNG THỰC TẾ VÀO TEMPLATE**

Có dữ liệu rồi thì có rất rất nhiều ứng dụng, một trong những ứng dụng là gán những những dữ liệu này vào một template và tạo các file cấu hình.

Dưới đây ví dụ là một file mà một kỹ sư mạng thường chuẩn bị để copy và paste vào thiết bị. File này thường được chuẩn bị bằng notepad cùng với các thao tác thủ công copy, paste và sửa chỗ khác biệt giữa các cổng, giữa các vlan và giữa các switch khác nhau. Sau khi làm xong các file cấu hình thường là dò từ trên xuống dưới, dò từ dưới lên trên xem có chỗ nào sai sót không. Việc làm này thường mất rất nhiều thời gian và nhiều công sức nhưng vẫn không tránh khỏi sai sót (gõ nhầm vlan, gõ thiếu vlan, gán nhầm vlan lên cổng, khai báo thiếu vlan trên trunk, khai báo dư vlan trên trunk,…). Việc làm trên 1 hay đến 10 switch thì cũng xem như không có vấn đề gì, nhưng nếu làm 100 hay 200 switch thì đó là một vấn đề hoàn toàn khác, làm sao mà không sai được chứ? Cần phải có cách nào để làm việc này một cách tự động? Câu trả lời là dùng template.

vlan 10

name data

vlan 20

vlan voice

vlan 30

name DMZ

interface Fa0/1

switchport mode access

switchport access vlan 10

spanning-tree portfast

interface Fa0/2

switchport mode access

switchport access vlan 20

spanning-tree portfast

interface Fa0/3

switchport mode access

switchport access vlan 30

spanning-tree portfast

interface Fa0/23

switchport mode trunk

switch trunk allowed vlan 10,20,30

interface Fa0/24

switchport mode trunk

switch trunk allowed vlan 10,20,30

Để làm một template thì cần nhìn dưới góc độ lập trình sẽ thấy có những dòng lệnh lặp đi lặp lại, hoặc những dòng lệnh lặp lại như chỉ một thông số khác nhau.

Ví dụ nữa các lệnh sau cứ lặp đi lặp lại mà chỉ khác nhau ở vlan id và vlan name. Cần thiết phải xây dựng template cho cấu hình bên dưới.

vlan 10

name data

vlan 20

vlan voice

vlan 30

name DMZ

Những phần không thay đổ thì được giữa lại trong template, những phần thay đổi gọi là biến (variable) trong lập trình. Để mô tảm một biến biến trong jinja2 thì dùng {{ }} để báo cho jinja2 là đang khai báo biến. Bên trong là tên biến.

vlan {{vlan\_id}}

name {{vlan\_name}}

Nhưng vậy cũng chưa đủ vì có nhiều giá trị của vlan\_id và vlan\_name khi cần tạo hàng loạt các vlan. Nghĩ ngay đến vòng lặp for trong lập trình. Jinja2 cũng hỗ trợ khai báo vòng lặp for. Vòng lặp for được đóng trong {% %} và cần phải mô ra điểm kết thúc {% endfor %}

{% for vlan in vlans %}

vlan {{ vlan.id }}

name {{ vlan.name }}

{% endfor %}

Đoạn code trong jinja2 trên mô tả: duyệt danh sách các vlan trong một list tên là vlans, mỗi lần duyệt qua một phần tử nào trong vlans list thì gán giá trị cho nó vào biến tạm vlan, rồi lại truy cập giá trị id và name của biến tạm vlan đó và gán vào template

Làm tương tự khi duyệt danh sách các interfaces. Bên trong vòng lặp for duyệt các interfaces thì có xét điều kiện nếu như cổng là access thì dùng tập lệnh liên quan đến cổng access, còn nếu là cổng trunk thì dùng tập lệnh liên quan đến cổng trunk.

{%for interface in interfaces%}

{%if interface.mode == 'access'%}

interface {{interface.id}}

switchport access vlan {{ interface.vlan }}

spanning-tree portfast

{%endif%}

{%if interface.mode == 'trunk'%}

interface {{interface.id}}

switchport mode trunk

switchport trunk allowed-vlan {{interface.vlan\_list}}

{%endif%}

{%endfor%}

Câu hỏi đặt ra lúc này là vlans và interfaces để dùng trong vòng lặp for lấy ở đâu? Làm sao jinja2 template có những giá trị này? Câu trả lời là python script sẽ truyền những giá trị này đến cho nó.

import yaml

from jinja2 import Environment,FileSystemLoader

with open('vlan.yaml', 'r') as f:

data = yaml.safe\_load(f.read())

file\_loader = FileSystemLoader('.')

env = Environment(loader=file\_loader)

template = env.get\_template('vlan.j2')

for device, config in data.items():

output = template.render(vlans=config['vlans'], interfaces=config['interfaces'])

with open(f'{device}.txt', 'w') as f:

f.write(output)

Đoạn code trên mô tả cách lấy dữ liệu vlans và interfaces trong dictionary là truyền vào các biến vlans và interfaces thông qua hàm render. Qua đó, jinja2 template đón nhận các giá trị này và chạy các vòng lặp for của chính nó.

Kết quả chạy đoạn code trên là các file SW1.txt, SW2.txt… được tự động sinh ra với những cấu hình tương ứng. Các file này có thể dùng bằng cách gửi cho kỹ sư kết nối cổng console/telnet/ssh, copy và paste vào, hoặc có thể dùng cách gửi cấu hình đến thiết bị từ python bằng module Netmiko (với điều kiện thiết bị có IP và có thể truy cập từ xa bằng telnet hoặc SSH từ máy chạy python script này)

vlan 10

name data

vlan 20

name voice

vlan 30

name DMZ

interface Fa0/1

switchport access vlan 10

spanning-tree portfast

interface Fa0/2

switchport access vlan 20

spanning-tree portfast

interface Fa0/3

switchport access vlan 30

spanning-tree portfast

interface Fa0/23

switchport mode trunk

switchport trunk allowed-vlan 10,20,30

interface Fa0/24

switchport mode trunk

switchport trunk allowed-vlan 10,20,30

**CÁC FILE HOÀN CHỈNH**

vlan.yaml

SW1:

vlans:

- id: 10

name: data

- id: 20

name: voice

- id: 30

name: DMZ

interfaces:

- id: Fa0/1

vlan: 10

mode: access

- id: Fa0/2

vlan: 20

mode: access

- id: Fa0/3

vlan: 30

mode: access

- id: Fa0/23

mode: trunk

vlan\_list: 10,20,30

- id: Fa0/24

mode: trunk

vlan\_list: 10,20,30

SW2:

vlans:

- id: 10

name: data

mode: access

- id: 20

name: voice

mode: access

interfaces:

- id: Fa0/1

vlan: 10

mode: access

- id: Fa0/2

vlan: 10

mode: access

- id: Fa0/3

vlan: 10

mode: access

- id: Fa0/4

vlan: 20

mode: access

- id: Fa0/23

mode: trunk

vlan\_list: 10,20,30

- id: Fa0/24

mode: trunk

vlan\_list: 10,20,30

vlan.j2

{% for vlan in vlans %}

vlan {{ vlan.id }}

name {{ vlan.name }}

{% endfor %}

{%for interface in interfaces%}

{%if interface.mode == 'access'%}

interface {{interface.id}}

switchport access vlan {{ interface.vlan }}

spanning-tree portfast

{%endif%}

{%if interface.mode == 'trunk'%}

interface {{interface.id}}

switchport mode trunk

switchport trunk allowed-vlan {{interface.vlan\_list}}

{%endif%}

{%endfor%}

vlan\_creation.py

import yaml

from jinja2 import Environment,FileSystemLoader

with open('vlan.yaml', 'r') as f:

data = yaml.safe\_load(f.read())

file\_loader = FileSystemLoader('.')

env = Environment(loader=file\_loader)

template = env.get\_template('vlan.j2')

for device, config in data.items():

output = template.render(vlans=config['vlans'], interfaces=config['interfaces'])

with open(f'{device}.txt', 'w') as f:

f.write(output)