**TextFSM**

TextFSM là gì và tại sao nó lại quan trọng

+ Khi bạn dùng lệnh CLI trên thiết bị mạng thì output nó hiện ra là một chuỗi semi-constructed data

+ TextSFM là một module của Python cho phép phân tích cú pháp dữ liệu văn bản bán cấu trúc bằng Templates(Mẫu dữ liệu). Ban đầu, nó được phát triển để giúp kỹ sư mạng tự động hóa quá trình phân tích và trích xuất thông tin từ đầu ra qua các lệnh CLI của các thiết bị mạng.

+ TextFSM nhận hai đầu vào - một tệp mẫu và một tệp văn bản (như đầu ra CLI) và trả về một danh sách các bản ghi chứa dữ liệu được phân tích từ văn bản đó.

Ứng dụng của TextFSM

from netmiko import ConnectHandler

R1 = {

    'device\_type': 'cisco\_ios',

    'ip' : '192.168.1.62',

    'username' : 'admin',

    'password' : 'netmiko',

    'secret' : 'cisco',

}

connect = ConnectHandler(\*\*R1)

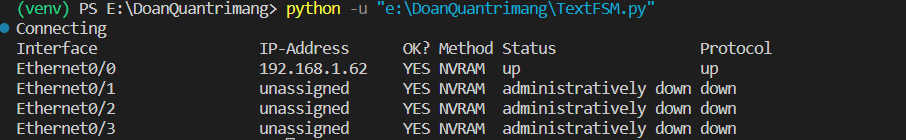
print('Connecting')

connect.enable()

output = connect.send\_command('sh ip int br')

print(output)

+ Đoạn code này chưa dung bất kì biện pháp phân tích cú pháp nào



(Đầu ra của đoạn code đó khi chưa có bất kì phân tích cú pháp nào)

from netmiko import ConnectHandler

import json

R1 = {

    'device\_type': 'cisco\_ios',

    'ip' : '192.168.1.62',

    'username' : 'admin',

    'password' : 'netmiko',

    'secret' : 'cisco',

}

connect = ConnectHandler(\*\*R1)

print('Connecting')

connect.enable()

output = connect.send\_command('sh ip int br', use\_textfsm=True)

print(json.dumps(output, indent=4))

+ Đoạn Code trên đã sử dụng thêm thư viện json và thêm use\_textfsm khi gửi lệnh ‘sh ip int br’

+ Kết quả in ra sau khi được phân tích và và kết hợp lại dễ dàng hơn cho ta điều khiển



(Kết quả đầu ra sau khi đã được phân tích cú pháp)

Bây giờ ta sẽ thử in ra những interface đang ở trạng thái administratively down

from netmiko import ConnectHandler

import json

R1 = {

    'device\_type': 'cisco\_ios',

    'ip' : '192.168.1.62',

    'username' : 'admin',

    'password' : 'netmiko',

    'secret' : 'cisco',

}

connect = ConnectHandler(\*\*R1)

print('Connecting')

connect.enable()

output = connect.send\_command('sh ip int br', use\_textfsm=True)

not\_connect\_interfaces = [item['intf'] for item in output if item['status'] == 'administratively down' ]

print(not\_connect\_interfaces)

+ Đoạn code trên đã thêm biến not\_connect\_interfaces lưu những interface đang ở trạng thái administratively down và in ra

A picture containing text, screenshot, font, line

Description automatically generated

(Kết quả sau khi in)

Cấu hình thiết bị bằng Key của TextFSM

from netmiko import ConnectHandler

import json

R1 = {

    'device\_type': 'cisco\_ios',

    'ip' : '192.168.1.62',

    'username' : 'admin',

    'password' : 'netmiko',

    'secret' : 'cisco',

}

connect = ConnectHandler(\*\*R1)

print('Connecting')

connect.enable()

output = connect.send\_command('sh ip int br', use\_textfsm=True)

not\_connect\_interfaces = [item['intf'] for item in output if item['status'] == 'administratively down' ]

for interface in not\_connect\_interfaces:

    commands = [f"interface {interface}", 'no shutdown']

    config\_output = connect.send\_config\_set(commands)

    print(config\_output)

print(connect.send\_command('sh ip int br'))

+ Ta dung vòng for để gửi lệnh đến từng thiết bị và thay vì nhập tay các interface, ta có thể dung value của Key interface đã lấy được ở trên để gửi một cách tự động.

+ Biến config\_output sẽ lưu lại các lệnh đã thực hiện ở mode config của interface đó

A screen shot of a computer program

Description automatically generated with medium confidence

+ In đầu ra của lệnh ‘sh ip int br’ để kiểm tra

A screenshot of a computer

Description automatically generated with medium confidence