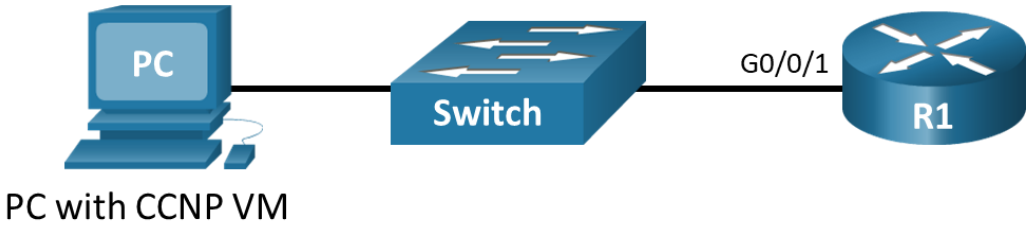
Lab: Sử dụng RESTCONF kết nối Router IOS XE

Sơ đồ



Bảng địa chỉ

Device	Interface	IP Address	Subnet Mask
R1	G0/0/1	192.168.1.1	255.255.255.0
PC	NIC	DHCP	DHCP

Mục tiêu

Part 1: Xây dựng mạng và kiểm tra kết nối

Part 2: Cấu hình thiết bị IOS XE cho kết nối RESTCONF

Part 3: Mở và cấu hình Postman

Part 4: Sử dụng Postman để gửi yêu cầu GET

Part 5: Sử dụng Postman để gửi yêu cầu PUT

Part 6: Dùng Python Script để gửi yêu cầu GET

Part 7: Dùng Python Script để gửi yêu cầu PUT

Bối cảnh/ Tổng quan

Giao thức RESTCONF là một tập hợp con của NETCONF. RESTCONF cho phép thực hiện các cuộc gọi API RESTful đến thiết bị IOS XE. Dữ liệu được API trả về có thể là XML hoặc JSON. Trong nửa đầu của lab này, chúng ta sẽ sử dụng chương trình Postman để xây dựng và gửi các yêu cầu API đến dịch vụ RESTCONF đang chạy trên R1. Trong nửa sau của lab, chúng ta sẽ tạo các script Python để thực hiện các tác vụ giống như Postman.

Các tài nguyên cần thiết

- 1 Router (Cisco 4221 with Cisco IOS XE Release 16.9.4 universal image or comparable)
- 1 Switch
- 1 PC (Choice of operating system with Cisco Networking Academy CCNP VM running in a virtual machine client and terminal emulation program)
- Cáp Ethernet

Hướng dẫn

Part 1: Xây dựng mạng và kiểm tra kết nối

Trong Phần 1, bạn sẽ nối cáp các thiết bị, khởi động Python VM và cấu hình R1 để truy cập NETCONF và RESTCONF qua kết nối SSH. Sau đó, bạn sẽ xác minh kết nối giữa Python VM và R1, kiểm tra kết nối SSH với R1.

Step 1: Cắm dây cho mạng như trên sơ đồ.

Kết nối các thiết bị như thể hiện trong sơ đồ cấu trúc liên kết và cáp nếu cần.

Step 2: Mở CCNP VM.

Ghi chú: Nếu bạn chưa hoàn thành Lab - Cài đặt Máy ảo CCNP, hãy thực hiện ngay trước khi tiếp tục với lab này.

a. Mở VirtualBox. Bắt đầu máy ảo **CCNP VM**.

b. Nhập mật khẩu **StudentPass** để đăng nhập vào máy ảo nếu cần.

Step 3: Cấu hình R1.

Kết nối với màn hình điều khiển của R1 và cấu hình các cài đặt cơ bản cho lab như sau:

```
enable
configure terminal
hostname R1
no ip domain lookup
line con 0
logging synchronous
 exec-timeout 0 0
line vty 0 15
 exec-t 0 0
 logg sync
 login local
 transport input ssh
!Configure SSH which is required for NETCONF and RESTCONF access
ip domain name example.netacad.com
crypto key generate rsa modulus 2048
username cisco privilege 15 password cisco123!
interface GigabitEthernet0/0/1
 description Link to PC
 ip address 192.168.1.1 255.255.255.0
 no shutdown
!Configure a DHCP server to assign IPv4 addressing to the CCNP VM
ip dhcp excluded-address 192.168.1.1 192.168.1.10
ip dhcp pool LAN
 network 192.168.1.0 /24
 default-router 192.168.1.1
 domain-name example.netacad.com
```

```
end
copy run start
```

Step 4: Kiểm tra CCNP VM có thể ping default gateway.

a. Trong **CCNP VM**, mở cửa sổ terminal.

b. Kiểm tra **CCNP VM** được kết nối với R1 bằng cách nhập địa chỉ IP để xác minh rằng CCNP VM đã nhận địa chỉ IP từ máy chủ DHCP hoặc bằng cách ping R1 ở 192.168.1.1. Nhập Ctrl + C để thoát ra khỏi ping, như thể hiện trong ví dụ đầu ra bên dưới.

```
student@Ubuntu-Master:~$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:50:56:b3:72:3b brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.15/24 brd 192.168.1.255 scope global dynamic noprefixroute ens160
       valid_lft 79564sec preferred_lft 79564sec
    inet6 fe80::1ae4:952f:402d:6b1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 00:50:56:b3:26:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.183/24 brd 192.168.50.255 scope global dynamic noprefixroute
ens192
       valid_lft 70687sec preferred_lft 70687sec
    inet6 fe80::4c87:a2b3:aa9:5470/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

student@Ubuntu-Master:~$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=255 time=0.703 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=255 time=0.748 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=255 time=0.757 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.703/0.736/0.757/0.023 ms
```

c. Nếu máy ảo CCNP chưa nhận được địa chỉ IPv4, hãy kiểm tra các kết nối vật lý của bạn giữa PC và R1. Ngoài ra, xác minh rằng R1 được cấu hình đúng theo bước trước đó.

Step 5: Thiết lập kết nối SSH đến R1.

a. Mở PuTTY SSH Client.

b. Nhập địa chỉ IPv4 cho default gateway, 192.168.1.1, và nhấn **Open**.

c. Bạn sẽ có thể đăng nhập vào R1 bằng tên người dùng cisco và mật khẩu cisco123!. Nếu không, hãy xác minh rằng cấu hình SSH của bạn là chính xác trên R1.

Part 2: Cấu hình thiết bị IOS XE để truy cập RESTCONF

Trong Phần 2, bạn sẽ cấu hình R1 để chấp nhận các thông báo RESTCONF. Bạn cũng sẽ bắt đầu dịch vụ HTTPS trên R1.

Step 1: Xác minh RESTCONF đang chạy trên R1.

RESTCONF có thể đã chạy nếu một học viên khác kích hoạt nó hoặc nếu phiên bản iOS mới hơn kích hoạt theo mặc định. Từ thiết bị đầu cuối PuTTY, bạn có thể sử dụng lệnh **show platform software yang-management process** để xem liệu tất cả các trình tiện ích được liên kết với dịch vụ RESTCONF có đang chạy không.

```
R1# show platform software yang-management process
confd            : Not Running
nesd             : Not Running
syncfd           : Not Running
ncsshd           : Not Running
dmiauthd         : Not Running
nginx            : Not Running
ndbmand          : Not Running
pubd             : Not Running
```

Ghi chú: Mục đích và chức năng của tất cả các trình tiện ích nằm ngoài phạm vi của khóa học này.

Step 2: Bật và kiểm tra dịch vụ RESTCONF.

a. Nhập lệnh cấu hình restconf để kích hoạt dịch vụ RESTCONF trên R1.

```
R1(config)# restconf
```

b. Xác minh rằng các trình tiện ích RESTCONF cần thiết hiện đang chạy. Hãy nhớ rằng ncsshd là dịch vụ NETCONF, có thể đang chạy trên thiết bị của bạn. Chúng tôi không cần nó cho lab này. Tuy nhiên, bạn cần nginx, đó là máy chủ HTTPS. Điều này sẽ cho phép thực hiện các cuộc gọi REST API đến dịch vụ RESTCONF.

```
R1(config)# exit
R1# show platform software yang-management process
confd            : Running
nesd             : Running
syncfd           : Running
ncsshd           : Not Running
dmiauthd         : Running
nginx            : Not Running
ndbmand          : Running
pubd             : Running
```

Step 3: Bật và kiểm tra dịch vụ HTTPS .

a. Nhập các lệnh cấu hình toàn cầu sau để kích hoạt server HTTPS và chỉ định rằng xác thực máy chủ nên sử dụng cơ sở dữ liệu cục bộ.

```
R1# configure terminal
R1(config)# ip http secure-server
```

```
R1(config)# ip http authentication local
```

b. Kiểm tra server HTTPS (nginx) đang chạy.

```
R1(config)# exit
R1# show platform software yang-management process
confd            : Running
nesd             : Running
syncfd           : Running
ncsshd           : Not Running
dmiauthd         : Running
nginx            : Running
ndbmand          : Running
pubd             : Running
```

Part 3: Mở và cấu hình Postman

Trong Phần 3, bạn sẽ mở Postman, vô hiệu hóa chứng chỉ SSL và khám phá giao diện người dùng.

Step 1: Mở Postman.

a. Trong máy ảo **CCNP VM**, mở ứng dụng Postman.

b. Khi mới mở Postman lần đầu, nó sẽ yêu cầu bạn tạo một tài khoản hoặc đăng nhập. Ở dưới cùng của cửa sổ, bạn cũng có thể nhấp vào dòng Skip để bỏ qua đăng nhập. Không cần đăng nhập để sử dụng ứng dụng này.

Step 2: Vô hiệu hóa chức năng xác thực chứng chỉ SSL.

Theo mặc định, Postman đã bật xác minh chứng nhận SSL. Bạn sẽ không sử dụng chứng chỉ SSL với R1; do đó, bạn cần tắt tính năng này.

a. Nhấn **File > Settings**.

b. Dưới tab **General**, chỉnh **SSL certificate verification thành OFF**.

c. Đóng **Settings** dialog box.

Part 4: Sử dụng Postman để gửi yêu cầu GET

Trong Phần 4, bạn sẽ sử dụng Postman để gửi yêu cầu GET đến R1 để xác minh rằng bạn có thể kết nối với dịch vụ RESTCONF.

Step 1: Khai thác giao diện người dùng.

a. Ở trung tâm, chúng ta sẽ thấy Launchpad. Bạn có thể khám phá khu vực này nếu muốn.

b. Nhấp vào dấu cộng (+) bên cạnh tab Launchpad để mở yêu cầu GET . Giao diện này là nơi bạn sẽ thực hiện tất cả công việc của mình trong lab này.

Step 2: Nhập URL cho R1.

a. Loại yêu cầu đã được đặt thành GET.

b. Trong trường request URL, nhập vào URL sẽ được sử dụng để truy cập dịch vụ RESTCONF đang chạy trên R1:

```
https://192.168.1.1/restconf/
```

Step 3: Nhập thông tin xác thực.

Bên dưới trường URL, sẽ có các tab như **Params**, **Authorization**, **Headers**, **Body**, **Pre-request Script**, and **Test**. Trong bài lab này, chúng ta sẽ sử dụng **Authorization**, **Headers**, và **Body**.

a. Nhấn tab **Authorization**.

b. Trong Type, nhấn dấu mũi tên xuống kế bên "Inherit auth from parent" và chọn **Basic Auth**.

c. Phần **Username** và **Password**, nhập thông tin xác thực đã được cáu hình trên R1 ở Part 1:

Username: **cisco**

Password: **cisco123!**

Step 4: Thêm thông tin xác thực vào request headers.

a. Nhân nút **Preview Request**. Điều này sẽ thêm thông tin xác thực vào tiêu đề yêu cầu.

b. Nhấn **Headers**. Sau đó mở rộng trường **Temporary Headers**. Chúng ta có thể xác minh rằng Authorization key có giá trị Basic sẽ được sử dụng để xác thực yêu cầu khi được gửi tới R1.

Step 5: Đặt JSON làm kiểu dữ liệu để gửi và nhận trên R1.

Chúng ta có thể gửi và nhận dữ liệu từ R1 ở định dạng JSON hoặc XML. Trong bài lab này, chúng ta sẽ dùng JSON.

a. Phía trên khu vực Temporary Headers là khu vực Headers. Nhấp vào trường Key và nhập Content-Type cho loại khóa. Trong trường Value, nhập application/ yang-data+json. Điều này bảo Postman gửi dữ liệu JSON đến R1.

b. Bên dưới **Content-Type**, thêm cặp key/value. Trường **Key** là **Accept** và trường **Value** là **application/yang-data+json**.

Ghi chú: Chúng ta có thể thay đổi application/yang-data+json thành application/yang-data+xml để gửi và nhận dữ liệu XML thay cho dữ liệu JSON nếu cần thiết.

Step 6: Gửi API request đến R1.

Postman bây giờ có tất cả thông tin cần thiết để gửi yêu cầu GET. Nhấp vào Send. Bên dưới Temporary Headers, chúng ta sẽ thấy phản hồi JSON sau đây từ R1. Nếu không, hãy xác minh rằng bạn đã hoàn thành các bước trước đó trong phần này của lab và cấu hình chính xác SSH trong Phần 1 và RESTCONF trong Part 2.

```
{
    "ietf-restconf:restconf": {
        "data": {},
        "operations": {},
        "yang-library-version": "2016-06-21"
    }
}
```

JSON response này xác minh rằng Postman hiện có thể gửi các yêu cầu REST API khác tới R1.

Step 7: Sử dụng GET request để thu thập thông tin của tất cả các cổng trên R1.

a. Bây giờ chúng ta đã có GET request thành công, bạn có thể sử dụng nó làm mẫu (template) cho các yêu cầu bổ sung. Ở đầu Postman, bên cạnh tab Launchpad, nhấp chuột phải vào tab GET mà bạn vừa sử dụng và chọn **Duplicate Current Tab**.

b. Sử dụng **ietf-interfaces** mô hình YANG để thu thập thông tin cổng. Trong URL, thêm **data/ietf-interfaces:interfaces**:

```
https://192.168.1.1/restconf/data/ietf-interfaces:interfaces
```

c. Nhấn **Send**. Chúng ta sẽ thấy JSON response từ R1 tương tự như đầu ra được hiển thị bên dưới. Đầu ra có thể khác nhau tùy thuộc vào router cụ thể.


```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet0",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": false,
        "ietf-ip:ipv4": {},
        "ietf-ip:ipv6": {}
      },
      {
        "name": "GigabitEthernet0/0/0",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": false,
        "ietf-ip:ipv4": {},
        "ietf-ip:ipv6": {}
      },
      {
        "name": "GigabitEthernet0/0/1",
        "description": "Link to PC",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "192.168.1.1",
              "netmask": "255.255.255.0"
            }
          ]
        },
        "ietf-ip:ipv6": {}
      }
    ]
  }
}
```


Step 8: Sử dụng GET request cho cổng chỉ định trên R1.

Trong lab này, chỉ cổng GigabitEthernet 0/0/0 là được cấu hình. Để chỉ định cổng này, mở rộng URL để chỉ nhận thông tin cổng.

a. Ghi lại GET request bước trước.

b. Thêm interface= tham số để chỉ định một cổng và nhập tên của cổng. Tuy nhiên, lưu ý rằng bạn cần sử dụng mã HTML% 2F cho các dấu gạch chéo về phía trước trong tên giao diện. Vì vậy, 0/0/1 trở thành 0%2F0%2F1.

```
https://192.168.1.1/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet0%2F0%2F1
```

Ghi chú: URL trên nên được ghi chung trên một dòng.

c. Nhấn **Send**. Chúng ta sẽ thấy phản hồi JSON từ R1 tương tự như đầu ra bên dưới. Đầu ra có thể khác nhau tùy thuộc vào router cụ thể.

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet0/0/1",
    "description": "Link to PC",
    "type": "iana-if-type:ethernetCsmacd",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "192.168.1.1",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

Part 5: Sử dụng Postman để gửi PUT Request

Trong Part 5, chúng ta sẽ cấu hình Postman để gửi PUT request đến R1 để tạo interface loopback mới.

Step 1: Sao y và sửa đổi yêu cầu GET cuối cùng.

Ghi chú: Cho bước này, sử dụng thông tin interface loopback được chỉ định bởi giảng viên. Nếu bạn không được chỉ định thông tin loopback và không có học viên nào khác truy cập vào router R1 cùng lúc với bạn, thì bạn có thể sử dụng thông tin loopback được chỉ định bên dưới.

a. Sao y yêu cầu GET cuối cùng.

b. Phần **Type** của request, nhấn nút mũi tên xuống kế bên **GET** và chọn **PUT**.

c. Phần tham số **interface=** , thay đổi nó thành **=Loopback1** để chỉ định cổng mới.

```
https://192.168.1.1/restconf/data/ietf-interfaces:interfaces/interface=Loopback1
```

Step 2: Định cấu hình phần thân của yêu cầu chỉ định thông tin cho loopback mới.

a. Để gửi yêu cầu PUT, chúng ta cần cung cấp thông tin phần thân của yêu cầu. Kế bên **Headers tab**, nhấn **Body**. Sau đó nhấn nút tròn **Raw**. Trường thông tin hiện tại đang trống. Nếu nhấn **Send** ngay, chúng ta sẽ nhận được status code **400 Bad Request** bởi vì Loopback1 chưa tồn tại và chúng ta chưa cung cấp đủ thông tin để tạo interface.

b. Thay thế văn bản trong ngoặc trong trường description bằng tên của bạn. Xóa dấu ngoặc. Điều này sẽ thêm tên của bạn vào mô tả cổng.

c. Điền vào phần Body với dữ liệu JSON cần thiết để tạo giao diện Loopback1 mới. Chúng ta có thể sao chép phần Body của yêu cầu GET trước đó và sửa đổi nó. Hoặc bạn có thể sao chép phần sau vào phần Body của yêu cầu PUT của bạn. Lưu ý rằng loại cổng phải được đặt thành **softwareLoopback**.

```
{
  "ietf-interfaces:interface": {
    "name": "Loopback1",
    "description": "[Student Name]'s Loopback",
    "type": "iana-if-type:softwareLoopback",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "10.1.1.1",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

d. Nhấp vào Send để gửi yêu cầu PUT đến R1. Trong trường phản hồi, bạn sẽ nhận được mã phản hồi HTTP được tạo 201. Điều này chỉ ra rằng tài nguyên đã được tạo thành công.

e. Bạn có thể xác minh rằng cổng đã được tạo bằng cách nhập show ip int brief trên R1. Bạn cũng có thể chạy tab Postman có chứa yêu cầu nhận thông tin về các cổng trên R1 đã được tạo trong Part 4 của lab này.

```
R1# show ip interface brief
Any interface listed with OK? value "NO" does not have a valid configuration

Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  administratively down down
GigabitEthernet0/0/1   192.168.1.1     YES manual up                    up
Serial0/1/0            unassigned      NO  unset  down                  down
Serial0/1/1            unassigned      NO  unset  down                  down
GigabitEthernet0       unassigned      YES unset  administratively down down
Loopback1              10.1.1.1        YES other  up                    up
```

Part 6: Sử dụng script Python để gửi GET Requests

Trong Part 6, Chúng ta sẽ viết script Python để gửi yêu cầu GET đến R1.

Step 1: Nhập mô-đun và tắt cảnh báo SSL.

a. Mở IDLE. Sau đó nhấn **File > New File** để mở IDLE Editor.

b. Lưu file tên **restconf-get.py**.

c. Nhập các lệnh sau để nhập các mô-đun được yêu cầu và tắt cảnh báo chứng chỉ SSL:

```
import json
import requests
```

```
requests.packages.urllib3.disable_warnings()
```

Mô-đun json bao gồm các phương thức để chuyển đổi dữ liệu JSON thành các đối tượng Python và ngược lại. Mô-đun yêu cầu có các phương thức cho phép chúng ta gửi các yêu cầu REST đến một URI.

Step 2: Tạo các biến sẽ là thành phần của yêu cầu.

Tạo một biến chuỗi để giữ URI điểm cuối API và hai từ điển, một cho các tiêu đề yêu cầu và một cho phần thân JSON. Lưu ý rằng đây là tất cả các nhiệm vụ giống như bạn đã hoàn thành trong ứng dụng Postman.

a. Tạo biến **api_url** và đăng ký URL sẽ kết nối thông tin cổng R1.

```
api_url = "https://192.168.1.1/restconf/data/ietf-interfaces:interfaces"
```

b. Tạo biến từ điển tên **headers** có keys là **Accept** và **Content-type** và đăng ký giá trị của keys là **application/yang-data+json**.

```
headers = { "Accept": "application/yang-data+json",
            "Content-type":"application/yang-data+json"
          }
```

c. Tạo một biến tuple Python có tên **basicauth** có hai khóa cần thiết để xác thực, **username** và **password**.

```
basicauth = ("cisco", "cisco123!")
```

Step 3: Tạo một biến để gửi yêu cầu và lưu trữ phản hồi JSON.

Sử dụng các biến được tạo ở bước trước làm tham số cho phương thức request.get (). Phương thức này gửi yêu cầu HTTP GET đến API RESTCONF trên R1. Gán kết quả của yêu cầu cho một biến có tên là resp. Biến đó sẽ giữ phản hồi JSON từ API. Nếu yêu cầu thành công, JSON sẽ chứa mô hình dữ liệu YANG được trả về.

a. Nhập câu lệnh sau:

```
resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
```

Bảng dưới đây liệt kê các yếu tố khác nhau của câu lệnh này:

Thuộc tính	Giải thích
`resp`	Biến giữ phản hồi từ API.
`requests.get()`	Phương thức để gửi yêu cầu GET.
`api_url`	Biến giữ chuỗi địa chỉ URL
`auth`	Biến tuple được tạo để giữ thông tin xác thực.
`headers=headers`	Một tham số được gán biến headers
`verify=False`	Tắt xác thực chứng chỉ SSL

b. Để xem HTTP response , thêm print.

```
print(resp)
```

c. Lưu và chạy script. Bạn sẽ nhận được đầu ra hiển thị dưới đây. Nếu không, hãy kiểm tra tất cả các bước trước trong phần này cũng như cấu hình SSH và RESTCONF cho R1.

```
================== RESTART: /home/student/restconf-script.py ==================
<Response [200]>
>>>
```

Step 4: Định dạng và hiển thị dữ liệu JSON nhận từ R1.

Bây giờ các giá trị phản hồi của mô hình YANG có thể được trích xuất từ response JSON.

a. JSON phản hồi không tương thích với từ điển Python và danh sách các đối tượng, vì vậy nó phải được chuyển đổi sang định dạng Python. Tạo một biến mới gọi là answer_json và gán biến tương ứng cho nó. Thêm phương thức json () để chuyển đổi JSON. Khai báo như sau:

```
response_json = resp.json()
```

b. Thêm lệnh print để hiện thị dữ liệu JSON.

```
print(response_json)
```

c. Lưu và chạy script. Chúng ta sẽ có thông tin đàu ra như sau:

```
================= RESTART: /home/student/restconf-script.py =================
<Response [200]>
{'ietf-interfaces:interfaces': {'interface': [{'name': 'GigabitEthernet0', 'type':
'iana-if-type:ethernetCsmacd', 'enabled': False, 'ietf-ip:ipv4': {}, 'ietf-ip:ipv6':
{}}, {'name': 'GigabitEthernet0/0/0', 'type': 'iana-if-type:ethernetCsmacd',
'enabled': False, 'ietf-ip:ipv4': {}, 'ietf-ip:ipv6': {}}, {'name':
'GigabitEthernet0/0/1', 'description': 'Link to PC', 'type': 'iana-if-
type:ethernetCsmacd', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip':
'192.168.1.1', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}}, {'name':
'Loopback1', 'description': "[Student's Name] Loopback", 'type': 'iana-if-
type:softwareLoopback', 'enabled': True, 'ietf-ip:ipv4': {'address': [{'ip':
'10.1.1.1', 'netmask': '255.255.255.0'}]}, 'ietf-ip:ipv6': {}}]}}
>>>
```

d. Để chỉnh sửa đầu ra, hãy chỉnh sửa câu lệnh in của bạn để sử dụng hàm json.dumps () với tham số indent:

```
print(json.dumps(response_json, indent=4))
```

e. Lưu và chạy script. Chúng ta sẽ nhận được đầu ra hiển thị dưới đây. Đầu ra này gần như giống hệt với đầu ra của yêu cầu GET Postman đầu tiên của bạn.

Ghi chú: Đầu ra sau đây bị cắt ngắn để chỉ hiển thị các cổng có cấu hình.

```
================= RESTART: /home/student/restconf-script.py =================
<Response [200]>
{
    "ietf-interfaces:interfaces": {
        "interface": [
(output omitted)
                "name": "GigabitEthernet0/0/1",
                "description": "Link to PC",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": true,
                "ietf-ip:ipv4": {
                    "address": [
                        {
                            "ip": "192.168.1.1",
                            "netmask": "255.255.255.0"
                        }
                    ]
                },
                "ietf-ip:ipv6": {}
```

```
            },
            {
                "name": "Loopback1",
                "description": "[Student's Name] Loopback",
                "type": "iana-if-type:softwareLoopback",
                "enabled": true,
                "ietf-ip:ipv4": {
                    "address": [
                        {
                            "ip": "10.1.1.1",
                            "netmask": "255.255.255.0"
                        }
                    ]
                },
                "ietf-ip:ipv6": {}
            }
        ]
    }
}
```

Part 7: Sử dụng Python Script để gửi PUT Request

Trong Phần 7, bạn sẽ tạo một script Python để gửi yêu cầu PUT đến R1. Như đã được thực hiện trong Postman, bạn sẽ tạo cổng loopback mới.

Step 1: Nhập mô-đun và tắt cảnh báo SSL.

Ghi chú: Đối với bước này, hãy sử dụng thông tin cổng loopback được chỉ định bởi người hướng dẫn của bạn. Tăng giá trị thêm 1 vào số cổng và octet thứ 2 của địa chỉ IP. Ví dụ: nếu bạn được chỉ định Loopback 200 và 10.200.1.1/24 để cấu hình địa chỉ loopback trong Postman, hãy sử dụng Loopback 201 và 10.201.1.1/24 cho bước này. Nếu bạn không được chỉ định thông tin loopback và không có học viên nào khác truy cập vào router R1 cùng lúc với bạn, thì có thể sử dụng thông tin loopback được chỉ định bên dưới.

a. Mở IDLE. Sau đó nhấn **File > New File** để mở IDLE Editor.

b. Lưu file tên **restconf-put.py**.

c. Nhập các lệnh sau để nhập mô-đun và tắt cảnh báo chứng chỉ SSL:

```
import json
import requests
requests.packages.urllib3.disable_warnings()
```

Step 2: Tạo các biến sẽ là thành phần của yêu cầu.

Tạo một biến string để giữ URI điểm cuối API và hai từ điển, một cho các tiêu đề yêu cầu và một cho phần thân JSON. Lưu ý rằng đây là tất cả các nhiệm vụ giống như bạn đã hoàn thành trong ứng dụng Postman.

a. Tạo một biến có tên api_url và gán cho nó URL với mục tiêu tạo cổng Loopback2 mới.

Ghi chú: Biến chỉ định này phải nằm trên một dòng trong script.

```
api_url = "https://192.168.1.1/restconf/data/ietf-interfaces:interfaces/interface=Loopback2"
```

b. Tạo một biến từ điển có tên là các header có các khóa cho Accecpt và Content-Type và gán giá trị cho các khóa là **application/yang-data+json**.

```
headers = { "Accept": "application/yang-data+json",
            "Content-type":"application/yang-data+json"
          }
```

c. Tạo một biến tuple Python có tên basicauth có hai giá trị cần thiết để xác thực, tên người dùng và mật khẩu.

```
basicauth = ("cisco", "cisco123!")
```

d. Tạo một biến từ điển Python yangConfig sẽ giữ dữ liệu YANG được yêu cầu để tạo cổng mới Loopback2. Bạn có thể sử dụng cùng một từ điển mà bạn đã sử dụng trong Part 5 cho Postman. Tuy nhiên, thay đổi số cổng và địa chỉ. Ngoài ra, hãy lưu ý rằng các giá trị Boolean phải được viết hoa trong Python. Do đó, đảm bảo rằng chữ T được viết hoa trong cặp khóa / giá trị cho **"enabled": True**.

```
yangConfig = {
    "ietf-interfaces:interface": {
        "name": "Loopback2",
        "description": "[Student\'s Name] loopback interface",
        "type": "iana-if-type:softwareLoopback",
        "enabled": True,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "10.2.1.1",
                    "netmask": "255.255.255.0"
                }
            ]
        },
        "ietf-ip:ipv6": {}
    }
}
```

Ghi chú: Nếu sử dụng dấu nháy đơn trong chuỗi văn bản mô tả, thì bạn phải thoát nó bằng dấu gạch chéo ngược, như được hiển thị ở trên.

Step 3: Tạo biến để gửi yêu cầu và lưu trữ JSON response.

Sử dụng các biến được tạo ở bước trước làm tham số cho phương thức request.put(). Phương thức này gửi yêu cầu PUT HTTP đến RESTCONF API. Gán kết quả của yêu cầu cho một biến có tên là resp. Biến đó sẽ giữ phản hồi JSON từ API. Nếu yêu cầu thành công, JSON sẽ chứa mô hình dữ liệu YANG được trả về.

a. Trước khi nhập các câu lệnh, xin lưu ý rằng đặc tả biến này chỉ nên ghi một dòng trong script. Nhập các lệnh sau:

Ghi chú: Giá trị của biến này phải nằm trên một dòng trong script của bạn.

```
resp = requests.put(api_url, data=json.dumps(yangConfig), auth=basicauth,
headers=headers, verify=False)
```

b. Nhập mã dưới đây để xử lý phản hồi. Nếu phản hồi là một trong những thông báo HTTP thành công, thông báo đầu tiên sẽ được in. Bất kỳ giá trị mã khác được coi là một lỗi. Mã phản hồi và thông báo lỗi sẽ được in trong trường hợp phát hiện lỗi.

```
if(resp.status_code >= 200 and resp.status_code <= 299):
    print("STATUS OK: {}".format(resp.status_code))
else:
    print('Error. Status Code: {} \nError message: {}'.format(resp.status_code,resp.json()))
```

Bảng dưới đây liệt kê các thuộc tính khác nhau của các trạng thái này

Thuộc tính	Giải thích
resp	Biến để giữ phản hồi từ API.
requests.put()	Phương thức thực sự tạo ra yêu cầu PUT.
api_url	Biến giữ chuỗi địa chỉ URL.
data	Dữ liệu sẽ được gửi đến điểm cuối API, được định dạng JSON.
auth	Biến tuple giữ thông tin xác thực
headers=headers	THông số được gắn để biểu hiện thông tin header.
verify=False	Thông số tắt xác thực SSL khi yêu cầu được gửi.
resp.status_code	Mã trạng thái HTTP trong trả lời yêu cầu API PUT.

c. Chạy tập lệnh để gửi yêu cầu PUT tới R1. Bạn sẽ nhận được một thông báo 201 Status Created. Nếu không, hãy kiểm tra mã của bạn và cấu hình cho R1.

d. Chúng ta có thể kiểm tra cổng đã tạo bằng lệnh **show ip interface brief** trên R1.

```
R1# show ip interface brief
Any interface listed with OK? value "NO" does not have a valid configuration

Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0/0   unassigned      YES unset  administratively down down
GigabitEthernet0/0/1   192.168.1.1     YES manual up                    up
Serial0/1/0            unassigned      NO  unset  down                  down
Serial0/1/1            unassigned      NO  unset  down                  down
GigabitEthernet0       unassigned      YES unset  administratively down down
Loopback1              10.1.1.1        YES other  up                    up
Loopback2              10.2.1.1        YES other  up                    up
```

Chương trình được sử dụng trong Lab

Scripts Python dưới đây được sử dụng trong lab này:

```
========================== resconf-get.py ==========================
import json
import requests
requests.packages.urllib3.disable_warnings()

api_url = "https://192.168.1.1/restconf/data/ietf-interfaces:interfaces"

headers = { "Accept": "application/yang-data+json",
            "Content-type":"application/yang-data+json"
            }

basicauth = ("cisco", "cisco123!")

resp = requests.get(api_url, auth=basicauth, headers=headers, verify=False)
```

```
print(resp)

response_json = resp.json()

print(json.dumps(response_json, indent=4))
```

```
========================= resconf-put.py =========================
import json
import requests
requests.packages.urllib3.disable_warnings()

api_url = "https://192.168.1.1/restconf/data/ietf-
interfaces:interfaces/interface=Loopback2"

headers = { "Accept": "application/yang-data+json",
            "Content-type":"application/yang-data+json"
           }
basicauth = ("cisco", "cisco123!")

yangConfig = {
    "ietf-interfaces:interface": {
        "name": "Loopback2",
        "description": "[Student\'s Name] loopback interface",
        "type": "iana-if-type:softwareLoopback",
        "enabled": True,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "10.2.1.1",
                    "netmask": "255.255.255.0"
                }
            ]
        },
        "ietf-ip:ipv6": {}
    }
}


resp = requests.put(api_url, data=json.dumps(yangConfig), auth=basicauth,
headers=headers, verify=False)

if(resp.status_code >= 200 and resp.status_code <= 299):
    print("STATUS OK: {}".format(resp.status_code))
else:
    print('Error. Status Code: {} \nError message: {}'.format(resp.status_code,
resp.json()))
```

Bảng tổng hợp cổng của Router

Router Model	Ethernet Interface #1	Ethernet Interface #2	Serial Interface #1	Serial Interface #2
1800	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
1900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2801	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
2811	Fast Ethernet 0/0 (F0/0)	Fast Ethernet 0/1 (F0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
2900	Gigabit Ethernet 0/0 (G0/0)	Gigabit Ethernet 0/1 (G0/1)	Serial 0/0/0 (S0/0/0)	Serial 0/0/1 (S0/0/1)
4221	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)
4300	Gigabit Ethernet 0/0/0 (G0/0/0)	Gigabit Ethernet 0/0/1 (G0/0/1)	Serial 0/1/0 (S0/1/0)	Serial 0/1/1 (S0/1/1)

Ghi chú: Để tìm hiểu cách cấu hình router, hãy xem các cổng để xác định loại router và số lượng cổng của router. Không có cách nào để liệt kê hiệu quả tất cả các kết hợp cấu hình cho từng lớp router. Bảng này bao gồm các mã định danh cho các kết hợp giao diện Ethernet và serial có thể có trong thiết bị. Bảng không bao gồm bất kỳ loại cổng nào khác, mặc dù một router cụ thể có thể chứa một loại cổng. Một ví dụ về điều này có thể là cổng ISDN BRI. Chuỗi trong ngoặc đơn là tên viết tắt hợp pháp có thể được sử dụng trong các lệnh Cisco IOS để thể hiện cổng.