Cisco Networking Academy® Mind Wide Open™

Práctica de laboratorio 12: Utilizar RESTCONF para acceder a un dispositivo IOS XE.

Versión en inglés:

https://itexam24.com/8-3-7-lab-use-restconf-to-access-an-ios-xe-device-answers/

Objetivos

Parte 1: Armar la red y verificar la conectividad.

Parte 2: Configurar un dispositivo IOS XE para el acceso RESTCONF.

Parte 3: Abrir y configurar Postman.

Parte 4: Usar Postman para enviar solicitudes GET.

Parte 5: Usar Postman para enviar solicitudes PUT.

Parte 6: Usar un script de Python para enviar solicitudes GET.

Parte 7: Usar un script de Python para enviar solicitudes PUT.

Aspectos básicos/Situación

El protocolo RESTCONF proporciona un subconjunto simplificado de características NETCONF sobre una API RESTful. RESTCONF nos permite realizar llamadas API RESTful a un dispositivo IOS XE. Los datos devueltos por la API pueden convertirse a formato XML o JSON. En la primera mitad de este laboratorio, utilizaremos el programa Postman para construir y enviar solicitudes API al servicio RESTCONF que se está ejecutando en el CSR1kV. En la segunda mitad del laboratorio, crearemos scripts de Python para realizar las mismas tareas que el programa Postman.

Recursos necesarios

- Una computadora con el sistema operativo de su elección.
- Virtual Box o VMWare.
- Máquina virtual DEVASC.
- Máquina Virtual CSR1kv

Instrucciones

Parte 1: Iniciar las máquinas virtuales y verificar la conectividad.

En esta parte, iniciaremos las dos máquinas virtuales del curso y verificaremos la conectividad. A continuación, estableceremos una conexión secure shell (SSH).

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

Paso 1: Iniciar las Máquinas Virtuales.

Si aún no ha completado el **Laboratorio - Instalar el entorno de laboratorio de máquina virtual** y el **Laboratorio - Instalar la VM CSR1kV**, complételos. Si ya ha completado estos laboratorios, inicie la máquina virtual DEVASC y la CSR1000v.

Paso 2: Verificar la conectividad entre las máquinas virtuales.

a. En la máquina virtual CSR1kv, presione Enter para poder empezar a ingresar comandos, luego use el comando **show ip interface brief** para verificar que la dirección IPv4 es 192.168.56.101. Si la dirección es diferente, anótela.

b. Abra el terminal en la máquina virtual DEVASC.

c. Haga ping hacia la CSR1kv para verificar la conectividad.

Ya debería haber hecho esto en los laboratorios de instalación. Si no puede hacer ping, vuelva a realizar los laboratorios mencionados anteriormente en el Paso 1 de la Parte 1.

```
devasc @labvm: ~$ ping -c 5 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=254 time=1.37 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=254 time=1.15 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=254 time=0.981 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=254 time=1.01 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=254 time=1.14 ms

--- 192.168.56.101 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.981/1.130/1.365/0.135 ms
devasc @labvm: ~$
```

Paso 3: Verificar la conectividad SSH a la máquina virtual CSR1kV.

a. En el terminal de la máquina virtual DEVASC, realice una conexión SSH hacia la máquina virtual CSR1kV con el siguiente comando:

```
devasc @ labvm: ~ $ ssh cisco@192.168.56.101
```

Nota: La primera vez que realizamos conexión SSH hacia la CSR1kV, la máquina virtual DEVASC nos advierte sobre la autenticidad del CSR1kV. Ya que confía en el CSR1kV, responda "yes" al prompt.

```
The authenticity of host '192.168.56.101 (192.168.56.101)' can't be
established.
RSA key fingerprint is SHA256:HYv9K5Biw7PFiXeoCDO/LTqs3EfZKBuJdiPo34VXDUY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.101' (RSA) to the list of known hosts.
```

b. Escriba **cisco123!** como contraseña y ahora debería estar en el modo EXEC privilegiado de la línea de comandos del CSR1kv.

```
Password: <cisco123!>

CSR1kv#
```

c. Deje abierta la sesión SSH, para la siguiente Parte.

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

www.netacad.com

Parte 2: Configurar un dispositivo IOS XE para el acceso RESTCONF.

En esta parte, configuraremos la máquina virtual CSR1kV para aceptar mensajes RESTCONF. Además, iniciaremos el servicio HTTPS.

Nota: Es posible que los servicios descritos en esta parte ya se estén ejecutando en su máquina virtual. Sin embargo, asegúrese de conocer los comandos para ver los servicios en ejecución y habilitarlos.

Paso 1: Comprobar que se estén ejecutando los daemons de RESTCONF.

RESTCONF ya debería estar en ejecución porque forma parte de la configuración predeterminada proporcionada por NetAcad.

Desde el terminal, puede utilizar el comando **show platform software yang-management process** para ver si se están ejecutando todos los daemons asociados al servicio RESTCONF. El daemon NETCONF también puede estar en ejecución, pero no se utilizará en este laboratorio. Si uno o más de los daemons requeridos no se están ejecutando, continúe con el paso 2.

```
CSR1kv# show platform software yang-management process
confd : Running
nesd: Running
syncfd: Running
ncsshd: Running
dmiauthd: Running
nginx: Running
ndbmand: Running
pubd: Running

CSR1kv#
```

Nota: El propósito y la función de todos los daemons está más allá del enfoque de este curso.

Paso 2: Habilitar y verificar el servicio RESTCONF.

a. Introduzca el comando de configuración global **restconf** para habilitar el servicio RESTCONF en el CSR1kV.

```
CSR1kv#configure terminal
CSR1kv(config)# restconf
```

b. Compruebe que los daemons RESTCONF necesarios se estén ejecutando ahora. Recuerde que **ncsshd** es el servicio NETCONF, que puede estar ejecutándose en su dispositivo. No lo necesitamos para este laboratorio. Sin embargo, necesita **nginx**, que es el servidor HTTPS. El daemon nginx le permitirá realizar llamadas API REST al servicio RESTCONF.

```
CSR1kv(config)# exit
CSR1kv# show platform software yang-management process
confd : Running
nesd: Running
syncfd: Running
ncsshd : Not Running
dmiauthd: Running
nginx : Not Running
ndbmand: Running
pubd: Running
```

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

Paso 3: Habilitar y verificar el servicio HTTPS.

a. Introduzca los siguientes comandos de configuración global para habilitar el servidor HTTPS y especificar que la autenticación del servidor debe utilizar la base de datos local.

```
CSR1kv# configure terminal
CSR1kv(config)# ip http secure-server
CSR1kv(config)# ip http authentication local
```

b. Compruebe que el servidor HTTPS (nginx) se esté ejecutando ahora.

```
CSR1kv(config)# exit
CSR1kv# show platform software yang-management process
confd : Running
nesd: Running
syncfd: Running
ncsshd : Not Running
dmiauthd: Running
nginx: Running
ndbmand: Running
pubd: Running
```

Parte 3: Abrir y configurar Postman.

En esta parte, abriremos Postman, deshabilitaremos los certificados SSL y exploraremos la interfaz de usuario.

Paso 1: Abrir Postman.

a. En la **VM DEVASC**, abra la aplicación Postman.

b. Si es la primera vez que abre Postman, es posible que le pida que cree una cuenta o inicie sesión. En la parte inferior de la ventana, también puede hacer clic en "Skip" (omitir) para omitir el inicio de sesión. No es necesario iniciar sesión para utilizar esta aplicación.

Paso 2: Deshabilitar la verificación de certificación SSL.

De forma predeterminada, Postman tiene activada la verificación de certificación SSL. No usaremos certificados SSL con el CSR1Kv, por lo tanto, debe desactivar esta función.

a. Haga clic en **Archivo > Configuración**.

b. En la pestaña **General**, establezca **SSL certificate verification** en **OFF**.

c. Cierre el cuadro de diálogo **Configuración**.

Parte 4: Utilizar Postman para enviar solicitudes GET.

En esta parte, utilizaremos Postman para enviar una solicitud GET al CSR1kV para verificar si podemos conectarnos al servicio RESTCONF.

Paso 1: Explorar la interfaz de usuario de Postman.

a. En el centro, verá el **Launchpad**. Puede explorar esta zona si lo desea.

b. Haga clic en el signo más (+) situado junto a la pestaña **Launchpad** para abrir una **Solicitud GET sin título**. Esta es la interfaz donde realizaremos todo el trabajo en este laboratorio.

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

Paso 2: Introduzca la dirección URL del CSR1kV.

a. El tipo de solicitud ya está establecido en GET. Deje el tipo de solicitud establecido en GET.

b. En el campo "Enter request URL" (Introducir URL de solicitud), escriba la URL que se utilizará para acceder al servicio RESTCONF que se está ejecutando en el CSR1kV:

```
https://192.168.56.101/restconf/
```

Paso 3: Introduzca las credenciales de autenticación.

En el campo URL, hay pestañas llamadas **Parámetros**, **Autorización**, **Encabezados**, **Contenido**, **Pre-request Script**, **Test**, y **Configuraciones**. En este laboratorio, utilizaremos **Autorización**, **Encabezados** y **Contenido**.

a. Haga clic en la pestaña **Autorización**.

b. En Tipo, haga clic en la flecha abajo situada junto a "Inherit auth from parent" (Heredar autenticación del administrador) y seleccione **Autenticación básica**.

c. En **Username** y **Password**, introduzca las credenciales de autenticación local para el CSR1kV:

Username: **cisco**

Password: **cisco123!**

d. Haga clic en **Encabezados**. Luego haga clic en **7 hidden** (7 ocultos). Podemos verificar que la Clave de autorización tiene un Valor básico que se utilizará para autenticar la solicitud cuando se envíe al CSR1kV.

Paso 4: Establezca JSON como el tipo de datos para enviar y recibir desde el CSR1kV.

Puede enviar y recibir datos desde el CSR1kv en formato XML o JSON. Para este laboratorio, utilizaremos JSON.

a. En el área **Encabezados**, haga clic en el primer campo **Clave** en blanco y escriba **Content-Type** para el tipo de clave. En el campo **Valor**, escriba **application/yang-data+json**. Esto le dice a Postman que envíe datos JSON al CSR1kV.

b. Debajo de la clave **Content-Type**, agregue otro par de clave/valor. El campo **Clave** es **Accept** y el campo **Valor** es **application/yang-data+json**.

Nota: Puede cambiar "application/yang-data+json" a "application/yang-data+xml" para enviar y recibir datos XML en lugar de datos JSON, si fuera necesario.

Paso 5: Enviar la solicitud de API al CSR1kV.

Postman ahora tiene toda la información que necesita para enviar la solicitud GET. Haga clic en **Enviar**. Debajo de **Encabezados temporales** debería poder ver la siguiente respuesta JSON del CSR1kV. Si no es así, compruebe que haya completado los pasos anteriores en esta parte del laboratorio y que haya configurado correctamente el servicio RESTCONF y HTTPS en la Parte 2.

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

Esta respuesta JSON verifica que Postman ahora puede enviar otras solicitudes de API REST al CSR1kV.

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

Paso 6: Utilizar una solicitud GET para recopilar la información de todas las interfaces del CSR1kv.

a. Ahora que tiene una solicitud GET exitosa, puede utilizarla como plantilla para solicitudes adicionales. En la parte superior de Postman, junto a la pestaña **Launchpad**, haga clic derecho en la pestaña **GET** que acaba de usar y seleccione **Duplicar pestaña**.

b. Utilice el modelo YANG de **ietf-interfaces** para recopilar información de la interfaz. En la URL, agregue **data/ietf-interfaces:interfaces**:

```
https://192.168.56.101/restconf/data/ietf-interfaces:interfaces
```

c. Haga clic en **Enviar**. Debería poder ver una respuesta JSON del CSR1kv que sea similar a la salida que se muestra a continuación. La salida puede ser diferente dependiendo del router que usted esté utlizando.

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "GigabitEthernet1",
        "description": "VBox",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-ip:ipv4": {},
        "ietf-ip:ipv6": {}
      }
    ]
  }
}
```

Paso 7: Utilizar una solicitud GET para recopilar información de una interfaz específica del CSR1kV.

En este laboratorio, solo la interfaz GigabitEthernet1 está configurada. Para especificar solo esta interfaz, extienda la URL para solicitar información para esta interfaz solamente.

a. Duplique su última solicitud GET.

b. Agregue el parámetro **interface=** para especificar una interfaz y escriba el nombre de la misma.

```
https://192.168.56.101/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1
```

Nota: Si solicita información de interfaz desde un dispositivo Cisco diferente con nombres que utilizan barras diagonales, como Gigabitethernet0/0/1, utilice el código HTML **%2F** para las barras diagonales en el nombre de la interfaz. De tal manera que, **0/0/1** se convierte en **0%2F0%2F1**.

c. Haga clic en **Enviar**. Debería poder ver una respuesta JSON del CSR1kv que sea similar a la salida que se muestra a continuación.

La salida puede ser diferente dependiendo del router que usted esté utilizando. En la configuración predeterminada de CSR1kv, no verá información de direccionamiento IP.

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet1",
    "description": "VBox",
    "type": "iana-if-type:ethernetCsmacd",
    "enabled": true,
```

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

```
    "ietf-ip:ipv4": {},
    "ietf-ip:ipv6": {}
  }
}
```

d. Esta interfaz recibe direcciones de una plantilla de Virtual Box. Por lo tanto, la dirección IPv4 no se muestra en **show running-config**. En su lugar, verá el comando **ip address dhcp**. Esto también se puede ver en la salida show ip interface brief.

```
CSR1kv# show ip interface brief
Interface IP-Address OK? Method Status Protocol
GigabitEthernet1 192.168.56.101 YES DHCP up up
CSR1kv#
```

e. En la siguiente Parte necesitará usar la respuesta JSON desde una interfaz configurada manualmente. Abra un terminal de comandos en el CSR1kv y configure manualmente la interfaz GigabitEthernet1 con la misma dirección IPv4 que fue asignada desde Virtual Box.

```
CSR1kv# conf t
CSR1kv(config)# interface g1
CSR1kv(config-if)# ip address 192.168.56.101 255.255.255.0
CSR1kv(config-if)# end
CSR1kv# show ip interface brief
Interface IP-Address OK? Method Status Protocol
GigabitEthernet1 192.168.56.101 YES manual up up
CSR1kv#
```

f. Regrese a Postman y envíe su solicitud GET de nuevo. Ahora debería ver información de direccionamiento IPv4 en la respuesta JSON, como se muestra a continuación.

En la siguiente Parte, copiará este formato JSON para crear una nueva interfaz.

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet1",
    "description": "VBox",
    "type": "iana-if-type:ethernetCsmacd",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "192.168.56.101",
          "netmask": "255.255.255.0"
        }
      ]
    },
    "ietf-ip:ipv6": {}
  }
}
```

Parte 5: Utilizar Postman para enviar una solicitud PUT.

En esta parte, configuraremos Postman para que envíe una solicitud PUT al CSR1kv para crear una nueva interfaz de loopback.

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

Nota: Si ha creado una interfaz loopback en otro laboratorio, quítela ahora o cree una nueva utilizando un número diferente.

Paso 1: Duplicar y modificar la última solicitud GET.

a. Duplique la última solicitud GET.

b. Para el **tipo** de solicitud, haga clic en la flecha abajo situada junto a **GET** y seleccione **PUT**.

c. Cambie el parámetro **interface=** por **=Loopback1** para especificar una nueva interfaz.

```
https://192.168.56.101/restconf/data/ietf-interfaces:interfaces/interface=Loopback1
```

Paso 2: Configurar el cuerpo de la solicitud especificando la información para el nuevo loopback.

a. Para enviar una solicitud PUT, debe proporcionar la información para el cuerpo de la solicitud. Junto a la pestaña **Encabezados**, haga clic en **Contenido**. Luego, haga clic en la opción **Raw**. El campo se encuentra vacío. Si hace clic en **Enviar**, obtendrá el código de error **400 Bad Request** porque Loopback1 aún no existe y no proporcionó suficiente información para crear la interfaz.

b. Rellene la sección **Contenido** con los datos JSON requeridos para crear una nueva interfaz Loopback1. Puede copiar la sección Contenido de la solicitud GET anterior y modificarla. O puede copiar lo siguiente en la sección Contenido de su solicitud PUT. Observe que el tipo de interfaz debe establecerse en **softwareLoopback**.

```
{
  "ietf-interfaces:interface": {
    "name": "Loopback1",
    "description": "My first RESTCONF 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": {}
  }
}
```

c. Haga clic en **Enviar** para enviar la solicitud PUT al CSR1kv. Debajo de la sección Contenido, debería poder ver el código de respuesta HTTP **Status: 201 Created**. Esto indica que el recurso se creó correctamente.

d. Puede verificar que se creó la interfaz. Vuelva a su sesión SSH con el CSR1kv e ingrese **show ip interface brief**. También puede ejecutar la pestaña Postman que contiene la solicitud para obtener información sobre las interfaces del CSR1kv que fue creada en la Parte anterior de este laboratorio.

```
CSR1kv# show ip interface brief
Interface IP-Address OK? Method Status Protocol
GigabitEthernet1 192.168.56.101 YES manual up up
Loopback1 10.1.1.1 YES other up up
CSR1kv#
```

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

Parte 6: Usar un script de Python para enviar solicitudes GET.

En esta Parte, crearemos un script de Python para enviar solicitudes GET al CSR1kv.

Paso 1: Crear el directorio RESTCONF y empezar a crear el script.

a. Abra VS Code. Luego haga clic en **Archivo > Abrir carpeta...** y diríjase al directorio **devnet-src**. Haga clic en **Aceptar**.

b. Abra una ventana de terminal en VS Code haciendo clic en: **Terminal> Nueva Terminal.**

c. Cree un subdirectorio que se llame **restconf** en el directorio **/devnet-src**.

```
devasc@labvm:~/labs/devnet-src$ mkdir restconf
devasc@labvm:~/labs/devnet-src$
```

d. En el panel **EXPLORER** en **DEVNET-SRC**, haga clic derecho en el directorio **restconf** y elija **Nuevo archivo**.

e. Nombre al archivo: **restconf-get.py**.

a. Introduzca los siguientes comandos para importar los módulos necesarios y deshabilitar las advertencias de certificado SSL:

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

El módulo **json** incluye métodos para convertir datos JSON a objetos Python y viceversa. El módulo **requests** tiene métodos que le permitirán enviar solicitudes REST a una URL.

Paso 2: Crear las variables que serán los componentes de la solicitud.

a. Cree una variable que se llame **api_url** y asígnele la URL que accederá a la información de la interfaz del CSR1kv.

```
api_url = "https://192.168.56.101/restconf/data/ietf-interfaces:interfaces"
```

b. Cree una variable tipo diccionario que se llame **encabezados**, dentro del diccionario cree las claves **Accept** y **Content-type** y asígneles el valor **application/yang-data+json**.

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

c. Cree una variable tipo tupla de Python que se llame **basicauth**, dentro de la tupla, cree las dos claves necesarias para la autenticación, **username** y **password**.

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

Paso 3: Crear una variable para enviar la solicitud y almacenar la respuesta JSON.

Utilice las variables que se crearon en el paso anterior como parámetros para el método **requests.get ()**. Este método envía una solicitud HTTP GET a la API RESTCONF del CSR1kv. Asigne el resultado de la solicitud a una variable que se llame **resp**. Esa variable contendrá la respuesta JSON de la API. Si la solicitud se realiza correctamente, el JSON contendrá el modelo de datos YANG devuelto.

a. Ingrese la siguiente instrucción:

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

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

En la siguiente tabla se enumeran los diversos elementos de esta instrucción

Elemento	Explicación
`resp`	La variable que va a contener la respuesta de la AP.
`requests.get()`	El método que realiza la solicitud GET.
`api_url`	La variable que contiene el string de la dirección URL.
`auth`	La variable tipo tupla creada para contener la información de autenticación.
`headers=headers`	Parámetro al que se le asigna la variable encabezados.
`verify=False`	Desactivar la verificación del certificado SSL cuando se realiza la solicitud.

b. Para ver el código de respuesta HTTP, agregue una instrucción print.

```
print(resp)
```

c. Guarde y ejecute el script. Debería obtener la salida que se muestra a continuación. Si no es así, compruebe todos los pasos anteriores de esta parte, así como la configuración SSH y RESTCONF del CSR1kv.

```
devasc@labvm:~/labs/devnet-src$ cd restconf/
devasc@labvm:~/labs/devnet-src/restconf$ python3 restconf-get.py
<Response [200]>
devasc@labvm:~/labs/devnet-src/restconf$
```

Paso 4: Formatear y mostrar los datos JSON recibidos del CSR1kV.

Ahora puede extraer los valores de respuesta del modelo YANG de la respuesta JSON.

a. El JSON de respuesta no es compatible con el diccionario ni con los objetos tipo lista de Python, por lo que debe convertirse al formato Python. Cree una nueva variable llamada **response_json** y asígnele la variable **resp**. Agregue el método **json ()** para convertir el JSON. La declaración es la siguiente:

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

b. Agregue una instrucción print para mostrar los datos JSON.

```
print(response_json)
```

c. Guarde y ejecute el script. Se debería obtener una salida similar a la siguiente:

```
devasc@labvm:~/labs/devnet-src/restconf$ python3 restconf-get.py
<Response [200]>
{'ietf-interfaces:interfaces': {'interface': [{'name': 'GigabitEthernet1',
'description': 'VBox', 'type': 'iana-if-type:ethernetCsmacd', 'enabled': True, 'ietf-
ip:ipv4': {'address': [{'ip': '192.168.56.101', 'netmask': '255.255.255.0'}]}, 'ietf-
ip:ipv6': {}}, {'name': 'Loopback1', 'description': 'My first RESTCONF 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': {}}]}}
devasc@labvm:~/labs/devnet-src/restconf$
```

d. Para embellecer la salida, edite la instrucción print para poder usar la función **json.dumps()** con el parámetro "indent" (sangría):

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

e. Guarde y ejecute el script. Debería obtener la salida que se muestra a continuación. Esta salida es prácticamente idéntica a la salida de su primera solicitud GET de Postman.

```
devasc@labvm:~/labs/devnet-src/restconf$ python3 restconf-get.py
```

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

```
<Response [200]>
{
    "ietf-interfaces:interfaces": {
        "interface": [
            {
                "name": "GigabitEthernet1",
                "description": "VBox",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": true,
                "ietf-ip:ipv4": {
                    "address": [
                        {
                            "ip": "192.168.56.101",
                            "netmask": "255.255.255.0"
                        }
                    ]
                },
                "ietf-ip:ipv6": {}
            },
            {
                "name": "Loopback1",
                "description": "My first RESTCONF 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": {}
            }
        ]
    }
}
devasc@labvm:~/labs/devnet-src/restconf$
```

Parte 7: Usar un script de Python para enviar una solicitud PUT.

En esta parte, crearemos un script de Python para enviar una solicitud PUT al CSR1kv. Al igual que se hizo en Postman, crearemos una nueva interfaz loopback.

Paso 1: Importar módulos y deshabilitar las advertencias SSL.

a. En el panel **EXPLORER** en **DEVNET-SRC**, haga clic derecho en el directorio **restconf** y elija **Archivo Nuevo**.

b. Asigne un nombre al archivo **restconf-put.py**.

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

c. Introduzca los siguientes comandos para importar los módulos necesarios y deshabilitar las advertencias de certificado SSL:

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

Paso 2: Crear las variables que serán los componentes de la solicitud.

a. Cree una variable llamada **api_url** y asígnele la URL que apunta a una nueva interfaz Loopback2.

Nota: Esta especificación de variable debe estar en una línea de su script.

```
api_url = "https://192.168.56.101/restconf/data/ietf-
interfaces:interfaces/interface=Loopback2"
```

b. Cree una variable tipo diccionario que se llame **encabezados** dentro del diccionario cree las claves Accept y Content-type y asígneles el valor **application/yang-data+json**.

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

c. Cree una variable tipo tupla de Python que se llame **basicauth**, dentro de la tupla, cree los dos valores necesarios para la autenticación, username y password.

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

d. Crear una variable tipo diccionario de Python que se llame **YangConfig** la cual contendrá los datos YANG que se requieren para crear la nueva interfaz Loopback2. Puede usar el mismo diccionario que utilizó anteriormente en Postman. Sin embargo, debe cambiar el número de interfaz y la dirección. Además, tenga en cuenta que los valores booleanos deben estar en mayúsculas en Python. Por lo tanto, asegúrese de que la **T** esté mayúscula en el par clave/valor de **"enabled": True**.

```
yangConfig = {
    "ietf-interfaces:interface": {
        "name": "Loopback2",
        "description": "My second RESTCONF loopback",
        "type": "iana-if-type:softwareLoopback",
        "enabled": True,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "10.2.1.1",
                    "netmask": "255.255.255.0"
                }
            ]
        },
        "ietf-ip:ipv6": {}
    }
}
```

Paso 3: Crear una variable para enviar la solicitud y almacenar la respuesta JSON.

Utilice las variables creadas en el paso anterior como parámetros para el método **requests.put ()**. Este método envía una solicitud PUT HTTP a la API RESTCONF. Asigne el resultado de la solicitud a una variable llamada **resp**. Esa variable contendrá la respuesta JSON de la API. Si la solicitud se realiza correctamente, el JSON contendrá el modelo de datos YANG devuelto.

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

a. Antes de introducir declaraciones, tenga en cuenta que esta especificación de variable debe estar en una sola línea de su script. Introduzca las siguientes instrucciones:

Nota: Esta especificación de variable debe estar en una línea de su script.

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

b. Introduzca el siguiente código para manejar la respuesta. Si la respuesta es uno de los mensajes "correcto o exitoso" 2XX de HTTP (HTTP success), se imprimirá el primer mensaje. Cualquier otro valor de código se considera un error. El código de respuesta y el mensaje de error se imprimirán en caso de que se haya detectado un error.

```
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()))
```

En la siguiente tabla se enumeran los diversos elementos de estas instrucciones:

Elemento	Explicación
`resp`	Variable para contener la respuesta de la API.
`requests.put()`	El método que realiza la solicitud PUT.
`api_url`	Variable que contiene el string de la dirección URL.
`data`	Los datos que se van a enviar al punto final de la API, los cuales están en formato JSON
`auth`	La variable tipo tupla creada para contener la información de autenticación
`headers=headers`	Parámetro al que se le asigna la variable headers
`verify=False`	Parámetro que desactiva la verificación de certificado SSL cuando se realiza la solicitud.
`resp.status_code`	El código de estado HTTP en la respuesta de la solicitud **PUT** de API.

c. Guarde y ejecute el script para enviar la solicitud PUT al CSR1kv. Debería recibir un mensaje que diga **201 Status Created**. Si no es así, compruebe su código y la configuración del CSR1kv.

d. Puede verificar que la interfaz fue creada, introduciendo **show ip interface brief** en el CSR1kv.

```
CSR1kv# show ip interface brief
Interface IP-Address OK? Method Status Protocol
GigabitEthernet1 192.168.56.101 YES manual up up
Loopback1 10.1.1.1 YES other up up
Loopback2 10.2.1.1 YES other up up
CSR1kv#
```

Programas utilizados en este laboratorio.

En este laboratorio se utilizaron los siguientes scripts de Python:

```
#=====================================================================
#resconf-get.py
import json
```

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

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

api_url = "https://192.168.56.101/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))

#end of file

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

api_url = "https://192.168.56.101/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": "My second RESTCONF loopback",
        "type": "iana-if-type:softwareLoopback",
        "enabled": True,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "10.2.1.1",
                    "netmask": "255.255.255.0"
                }
            ]
        },
```

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco

www.netacad.com

```
            "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()))

    #end of file
```

Fin del documento

© - 2023 Cisco y/o sus filiales. Todos los derechos reservados. Información pública de Cisco