Travaux pratiques - Utiliser RESTCONF pour accéder à un périphérique IOS XE

Objectifs

Partie 1 : Création du réseau et vérification de la connectivité

Partie 2 : Configurer un périphérique IOS XE pour l'accès RESTCONF

Partie 3 : Ouvrir et configurer Postman

Partie 4 : Utiliser Postman pour envoyer des requêtes GET

Partie 5: Utiliser Postman pour envoyer une demande de PUT

Partie 6 : Utiliser un script Python pour envoyer des requêtes GET

Partie 7 : Utiliser un script Python pour envoyer une requête PUT

Contexte/scénario

Le protocole RESTCONF fournit un sous-ensemble simplifié de fonctionnalités NETCONF sur une API RESTful. RESTCONF vous permet d'effectuer des appels d'API RESTful à un périphérique IOS XE. Les données renvoyées par l'API peuvent être formatées en XML ou en JSON. Dans la première moitié de ce laboratoire, vous utiliserez le programme Postman pour construire et envoyer des requêtes API au service RESTCONF qui s'exécute sur CSR1kv. Dans la seconde moitié du laboratoire, vous allez créer des scripts Python pour effectuer les mêmes tâches que votre programme Postman.

Ressources requises

- 1 PC avec système d'exploitation de votre choix
- Boîte virtuelle ou VMWare
- Machine virtuelle DEVASC
- Machine virtuelle CSR1kv

Instructions

Partie 1 : Lancez les machines virtuelles et vérifiez la connectivité

Dans cette partie, vous lancez les deux machines virtuelles pour le cours et vérifiez la connectivité. Vous établirez ensuite une connexion SSH (Secure Shell).

Étape 1: Lancez les machines virtuelles

Si vous n'avez pas encore terminé le **travail pratique - Installer l'environnement de laboratoire de machine virtuelle** et le **travail pratique - Installer la machine virtuelle CSR1kv**, faites-le maintenant. Si vous avez déjà terminé ces laboratoires, lancez dès maintenant la machine virtuelle DEVASC et la machine virtuelle CSR1000v.

Étape 2: Vérifiez la connectivité entre les machines virtuelles.

a. Dans la machine virtuelle CSR1kv, appuyez sur Entrée pour obtenir une invite de commande, puis utilisez **show ip interface brief** pour vérifier que l'adresse IPv4 est 192.168.56.101. Si l'adresse est différente, notez la.

b. Terminal ouvert dans le DEVASC VM.

c. Ping du CSR1kv pour vérifier la connectivité. Vous auriez déjà dû le faire dans les laboratoires d'installation. Si vous n'êtes pas en mesure de ping, consultez les laboratoires énumérés ci-dessus à la partie 1a.

```
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:~$
```

Étape 3: Vérifiez la connectivité SSH à la machine virtuelle CSR1kV.

a. Dans le terminal de la machine virtuelle DEVASC, SSH à la machine virtuelle CSR1kV avec la commande suivante :

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

Remarque : La première fois que vous SSH vers le CSR1kV, votre VM DEVASC vous avertit de l'authenticité du CSR1kV. Parce que vous faites confiance au CSR1kV, répondez oui à l'invite.

```
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. Entrez **cisco123!** comme mot de passe et vous devriez maintenant être à l'invite de commande EXEC privilégiée pour le CSR1kv.

```
Password: <cisco123!>


CSR1kv#
```

c. Laissez la session SSH ouverte pour la partie suivante.

Partie 2 : Configurer un périphérique IOS XE pour l'accès RESTCONF

Dans cette partie, vous allez configurer la machine virtuelle CSR1kv pour accepter les messages RESTCONF. Vous allez également démarrer le service HTTPS.

Remarque : Les services décrits dans cette partie sont peut-être déjà en cours d'exécution sur votre machine virtuelle. Cependant, assurez-vous de connaître les commandes permettant d'afficher les services en cours d'exécution et de les activer.

Étape 1: Vérifiez que les démons RESTCONF sont en cours d'exécution.

RESTCONF devrait déjà être en cours d'exécution car il faisait partie de la configuration par défaut fournie par NetACad. Depuis le terminal, vous pouvez utiliser la commande **show platform software yang-management process** pour voir si tous les démons associés au service RESTCONF sont en cours

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

d'exécution. Le démon NETCONF peut également être en cours d'exécution, mais il ne sera pas utilisé dans ce laboratoire. Si un ou plusieurs des démons requis ne sont pas en cours d'exécution, passez à l'étape 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#
```

Remarque : Le but et la fonction de tous les démons dépassent le cadre de ce cours.

Étape 2: Activez et vérifiez le service RESTCONF.

a. Entrez la commande de configuration globale **restconf** pour activer le service RESTCONF sur le CSR1kv.

```
CSR1kv#configure terminal
CSR1kV (config) # restconf
```

b. Vérifiez que les démons RESTCONF requis sont en cours d'exécution. Rappelez-vous que **ncsshd** est le service NETCONF, qui peut être exécuté sur votre appareil. On n'en a pas besoin pour ce TP. Cependant, vous avez besoin de **nginx**, qui est le serveur HTTPS. Cela vous permettra d'effectuer des appels d'API REST au service 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
```

Étape 3: Activez et vérifiez le service HTTPS.

a. Entrez les commandes de configuration globale suivantes pour activer le serveur HTTPS et spécifiez que l'authentification du serveur doit utiliser la base de données locale.

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

b. Vérifiez que le serveur HTTPS (nginx) est en cours d'exécution.

```
CSR1kv(config)# exit
CSR1kv# show platform software yang-management process
confd : Running
nesd : Running
```

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

```
syncfd : Running
ncsshd : Not Running
dmiauthd : Running
nginx : Running
ndbmand : Running
pubd : Running
```

Partie 3 : Ouvrir et configurer Postman

Dans cette partie, vous allez ouvrir Postman, désactiver les certificats SSL et explorer l'interface utilisateur.

Étape 1: Ouvrez Postman.

a. Dans la **DEVASC VM**,, ouvrez l'application Postman.

b. Si c'est la première fois que vous ouvrez Postman, il peut vous demander de créer un compte ou de vous connecter. En bas de la fenêtre, vous pouvez également cliquer sur le message « Skip » pour ignorer la connexion. Il n'est pas nécessaire de se connecter pour utiliser cette application.

Étape 2: Désactiver la vérification de la certification SSL.

Par défaut, Postman a activé la vérification de certification SSL. Vous n'utiliserez pas de certificats SSL avec le CSR1kV ; par conséquent, vous devez désactiver cette fonctionnalité.

a. Cliquez sur **File > Settings**.

b. Sous l'onglet **General**, définissez la **SSL certificate verification**sur **OFF**.

c. Fermez la boîte de dialogue **Settings**.

Partie 4 : Utiliser Postman pour envoyer des requêtes GET

Dans cette partie, vous utiliserez Postman pour envoyer une requête GET au CSR1kv afin de vérifier que vous pouvez vous connecter au service RESTCONF.

Étape 1: Explorez l'interface utilisateur de Postman.

a. Au centre, vous verrez le **Launchpad**. Vous pouvez explorer cette région si vous le souhaitez.

b. Cliquez sur le signe plus (+) en regard de l'onglet **Launchpad** pour ouvrir une **GET Untitled Request**. Cette interface est l'endroit où vous ferez tout votre travail dans ce laboratoire.

Étape 2: Entrez l'URL du CSR1kv.

a. Le type de demande est déjà défini sur GET. Laissez le type de demande défini sur GET.

b. Dans le champ « Entrer l'URL de requête », tapez l'URL qui sera utilisée pour accéder au service RESTCONF qui s'exécute sur le CSR1kv :

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

Étape 3: Saisissez les données d'authentification.

Sous le champ URL, des onglets sont répertoriés pour **Params**, **Authorization**, **Headers**, **Body**, **Pre-request Script**, **Test**et **Settings**. Dans ce TP, vous utiliserez **Authorization**, **Headers**, et **Body**.

a. Cliquez sur l'onglet **Authorization**.

b. Sous Type, cliquez sur la flèche vers le bas en regard de “Inherit auth from parent” et choisissez **Auth de base**.

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

c. Pour **Nom d'utilisateur** et **Mot de passe**, entrez les informations d'identification d'authentification locales pour le CSR1kv :

Username: **cisco**

Password: **cisco123!**

d. Cliquez sur **Headers**. Cliquez ensuite sur le **7 hidden**. Vous pouvez vérifier que la clé d'autorisation a une valeur de base qui sera utilisée pour authentifier la demande lorsqu'elle est envoyée au CSR1kv.

Étape 4: Définissez JSON comme type de données à envoyer et à recevoir à partir du CSR1kv.

Vous pouvez envoyer et recevoir des données à partir du CSR1kv au format XML ou JSON. Pour ce TP, vous utiliserez JSON.

a. Dans la zone **Headrers**, cliquez dans le premier champ **Key** vide et tapez **Content-Type** pour le type de clé. Dans le champ **Value**, tapez **application/yang-data+json**. Cela indique à Postman d'envoyer des données JSON au CSR1kV.

b. Sous votre clé **Content-Type**, ajoutez une autre paire clé/valeur. Le champ **Key** est **Accept** et le champ **Value** est **application/yang-data+json**.

Remarque : Vous pouvez changer application/yang-data+json en application/yang-data+xml pour envoyer et recevoir des données XML au lieu des données JSON, si nécessaire.

Étape 5: Envoyez la demande d'API au CSR1kv.

Le facteur a maintenant toutes les informations dont il a besoin pour envoyer la requête GET. Cliquez sur **Send** (Envoyer). Sous **Temporary Headers**, vous devriez voir la réponse JSON suivante du CSR1kv. Si ce n'est pas le cas, vérifiez que vous avez terminé les étapes précédentes dans cette partie du laboratoire et que vous avez correctement configuré le service RESTCONF et HTTPS dans la partie 2.

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

Cette réponse JSON vérifie que Postman peut maintenant envoyer d'autres requêtes d'API REST au CSR1kv.

Étape 6: Utilisez une requête GET pour collecter les informations pour toutes les interfaces sur le CSR1kv.

a. Maintenant que vous avez une requête GET réussie, vous pouvez l'utiliser comme modèle pour des demandes supplémentaires. En haut de Postman, à côté de l'onglet **Launchpad**, cliquez avec le bouton droit sur l'onglet **GET** que vous venez d'utiliser et choisissez **Duplicate Tab**.

b. Utilisez le modèle **ietf-interfaces** YANG pour collecter des informations sur l'interface. Pour l'URL, ajoutez **data/ietf-interfaces:interfaces**:

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

c. Cliquez sur **Send** (Envoyer). Vous devriez voir une réponse JSON du CSR1kV qui est similaire à la sortie montrée ci-dessous. Votre sortie peut être différente en fonction de votre routeur particulier.

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
```

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

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

Étape 7: Utilisez une requête GET pour collecter des informations pour une interface spécifique sur le CSR1kv.

Dans ce TP, seule l'interface GigabiteThernet1 est configurée. Pour spécifier uniquement cette interface, étendez l'URL pour demander uniquement des informations pour cette interface.

a. Dupliquez votre dernière requête GET.

b. Ajoutez le paramètre **interface=** pour spécifier une interface et tapez le nom de l'interface.

```
https://192.168.56.101/restconf/data/ietf-interfaces:interfaces/interface =GigabiteThernet1
```

Remarque : Si vous demandez des informations d'interface à partir d'un périphérique Cisco différent avec des noms qui utilisent des barres obliques, comme GigabiteThernet0/0/1, utilisez le code HTML **%2F** pour les barres obliques dans le nom de l'interface. Donc, **0/0/1** devient **0% 2F 0% 2F1**.

c. Cliquez sur **Send** (Envoyer). Vous devriez voir une réponse JSON du CSR1kV qui est similaire à la sortie ci-dessous. Votre sortie peut être différente en fonction de votre routeur particulier. Dans la configuration par défaut CSR1kv, vous ne verrez pas d'informations d'adressage IP.

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet1",
    "description": "VBox",
    "type": "iana-if-type:ethernetCsmacd",
    "activé": vrai,
    "ietf-ip:ipv4": {},
    "ietf-ip:ipv6": {}
  }
}
```

d. Cette interface reçoit l'adressage à partir d'un modèle de boîte virtuelle. Par conséquent, l'adresse IPv4 n'est pas affichée sous **show running-config**. Au lieu de cela, vous verrez la commande **ip address dhcp**. Vous pouvez le voir aussi dans la sortie courte de l'interface show ip.

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

e. Dans la partie suivante, vous devrez utiliser la réponse JSON à partir d'une interface configurée manuellement. Ouvrez un terminal de commande avec CSR1kv et configurez manuellement l'interface GigaBiteThernet1 avec la même adresse IPv4 qui lui est actuellement assignée par Virtual Box.

```
CSR1KV # conf t
CSR1kV (config) # interface g1
CSR1kv(config-if)# ip address 192.168.56.101 255.255.255.0
```

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

```
CSR1kv(config-if)# end
CSR1kv# show ip interface brief
Interface IP-Address OK? Method Status Protocol
GigabiteThernet1 192.168.56.101 OUI manuel up
CSR1kv#
```

f. Retournez à Postman et envoyez à nouveau votre demande GET. Vous devriez maintenant voir les informations d'adressage IPv4 dans la réponse JSON, comme indiqué ci-dessous. Dans la partie suivante, vous allez copier ce format JSON pour créer une nouvelle interface.

```
{
  "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": {}
  }
}
```

Partie 5 : Utiliser Postman pour envoyer une demande de PUT

Dans cette partie, vous allez configurer Postman pour envoyer une requête PUT au CSR1kv pour créer une nouvelle interface de bouclage.

Remarque : Si vous avez créé une interface de bouclage dans un autre laboratoire, supprimez-la maintenant ou créez-en une nouvelle en utilisant un numéro différent.

Étape 1: Dupliquer et modifier la dernière requête GET.

a. Dupliquez la dernière requête GET.

b. Pour le **type** de demande, cliquez sur la flèche vers le bas en regard de **GET** et choisissez **PUT**.

c. Pour le paramètre **interface=**, remplacez-le par **=Loopback1** pour spécifier une nouvelle interface.

```
https://192.168.56.101/restconf/data/ietf-interfaces:interfaces/Interface=LoopBack1
```

Étape 2: Configurez le corps de la demande en spécifiant les informations pour le nouveau bouclage.

a. Pour envoyer une demande PUT, vous devez fournir les informations pour le corps de la demande. À côté de l'onglet **Headers**, cliquez sur **Body**. Cliquez ensuite sur le bouton radio **Raw**. Le champ est actuellement vide. Si vous cliquez sur **Send** maintenant, vous obtiendrez le code d'erreur **400 Bad Request** car Loopback1 n'existe pas encore et vous n'avez pas fourni suffisamment d'informations pour créer l'interface.

b. Remplissez la section **Body** avec les données JSON requises pour créer une nouvelle interface Loopback1. Vous pouvez copier la section Corps de la requête GET précédente et la modifier. Vous

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

pouvez également copier ce qui suit dans la section Corps de votre demande PUT. Notez que le type d'interface doit être défini sur **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. Cliquez sur **Send** pour envoyer la demande PUT au CSR1kv. Sous la section Corps, vous devriez voir le code de réponse HTTP **Status: 201 Created**. Cela indique que la ressource a été créée avec succès.

d. Vous pouvez vérifier que l'interface a été créée. Retournez à votre session SSH avec le CSR1kv et entrez **show ip interface brief**. Vous pouvez également exécuter l'onglet Postman qui contient la demande d'obtenir des informations sur les interfaces sur le CSR1kv qui a été créé dans la partie précédente de ce laboratoire.

```
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#
```

Partie 6 : Utiliser un script Python pour envoyer des requêtes GET

Dans cette partie, vous allez créer un script Python pour envoyer des requêtes GET au CSR1kv.

Étape 1: Créez le répertoire RESTCONF et démarrez le script.

a. Ouvrez le code VS. Cliquez ensuite sur **File > Open Folder...** et accédez au répertoire **devnet-src**. Cliquez sur **OK**.

b. Ouvrez une fenêtre de terminal en VS Code : **Terminal > New Terminal**.

c. Créez un sous-répertoire appelé **restconf** dans le répertoire **/devnet-src**.

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

d. Dans le volet **EXPLORER** sous **DEVNET-SRC**, cliquez avec le bouton droit de la souris sur le répertoire **restconf** et choisissez **New File**.

e. Nommez le fichier **restconf-get.py**.

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

a. Entrez les commandes suivantes pour importer les modules requis et désactiver les avertissements de certificat SSL :

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

Le module **json** inclut des méthodes pour convertir des données JSON en objets Python et vice versa. Le module **requests** a des méthodes qui vous permettront d'envoyer des requêtes REST à une URL.

Étape 2: Créez les variables qui seront les composants de la demande.

a. Créez une variable nommée **api_url** et attribuez-lui l'URL qui accèdera aux informations de l'interface sur le CSR1kv.

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

b. Créez une variable de dictionnaire nommée **en-têtes** qui a des clés pour **Accept** et **Content-type** et attribuez aux clés la valeur **application/yang-data+json**.

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

c. Créez une variable tuple Python nommée **basicauth** qui a deux clés nécessaires pour l'authentification, **nom d'utilisateur** et **mot de passe**.

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

Étape 3: Créez une variable pour envoyer la demande et stocker la réponse JSON.

Utilisez les variables qui ont été créées à l'étape précédente comme paramètres pour la méthode **requests.get ()**. Cette méthode envoie une requête HTTP GET à l'API RESTCONF sur le CSR1kv. Affectez le résultat de la requête à une variable nommée **resp**. Cette variable contiendra la réponse JSON de l'API. Si la requête est réussie, le JSON contiendra le modèle de données YANG retourné.

a. Entrez l'instruction suivante :

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

Le tableau ci-dessous répertorie les différents éléments de cette déclaration :

Élément	Explication
`resp`	La variable pour contenir la réponse de l'API
`requests.get()`	La méthode qui fait réellement la requête GET
`api_url`	La variable qui contient la chaîne d'adresse URL
`auth`	La variable tuple créée pour contenir les informations d'authentification
`headers=headers`	Paramètre affecté à la variable en-têtes
`verify=False`	Désactive la vérification du certificat SSL lorsque la demande est faite

b. Pour afficher le code de réponse HTTP, ajoutez une instruction d'impression.

```
print(resp)
```

c. Sauvegardez et exécutez votre scénario. Vous devriez obtenir la sortie ci-dessous. Si ce n'est pas le cas, vérifiez toutes les étapes précédentes de cette partie ainsi que la configuration SSH et RESTCONF pour le CSR1kv.

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

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

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

Étape 4: Formatez et affichez les données JSON reçues du CSR1kv.

Maintenant, vous pouvez extraire les valeurs de réponse du modèle YANG à partir de la réponse JSON.

a. La réponse JSON n'est pas compatible avec les objets de dictionnaire Python et de liste, elle doit donc être convertie au format Python. Créez une nouvelle variable appelée **response_json** et attribuez-lui la variable **resp**. Ajoutez la méthode **json ()** pour convertir le JSON. La déclaration est la suivante :

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

b. Ajoutez une instruction print pour afficher les données JSON.

```
print (response_json)
```

c. Sauvegardez et exécutez votre scénario. Vous devriez obtenir un résultat similaire à ce qui suit :

```
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. Pour préttifier la sortie, modifiez votre instruction print pour utiliser la fonction **json.dumps ()** avec le paramètre "indent":

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

e. Sauvegardez et exécutez votre scénario. Vous devriez obtenir la sortie ci-dessous. Cette sortie est pratiquement identique à la sortie de votre première requête GET Postman.

```
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$
```

Partie 7 : Utiliser un script Python pour envoyer une requête PUT

Dans cette partie, vous allez créer un script Python pour envoyer une requête PUT au CSR1kv. Comme cela a été fait dans Postman, vous allez créer une nouvelle interface de bouclage.

Étape 1: Importez des modules et désactivez les avertissements SSL.

a. Dans le volet **EXPLORER** sous **DEVNET-SRC**, cliquez avec le bouton droit de la souris sur le répertoire **restconf** et choisissez **New File**.

b. Nommez le fichier **restconf-put.py**.

c. Entrez les commandes suivantes pour importer les modules requis et désactiver les avertissements de certificat SSL :

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

Étape 2: Créez les variables qui seront les composants de la demande.

a. Créez une variable nommée **api_url** et attribuez-lui l'URL qui cible une nouvelle interface Loopback2.

Remarque : Cette spécification de variable doit se trouver sur une ligne dans votre script.

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

b. Créez une variable de dictionnaire nommée **headers** qui a des clés pour Accept et Content-type et attribuez aux clés la valeur **application/yang-data+json**.

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

c. Créez une variable tuple Python nommée **basicauth** qui a deux valeurs nécessaires pour l'authentification, nom d'utilisateur et mot de passe.

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

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

d. Créez une variable de dictionnaire Python **YangConfig** qui contiendra les données YANG requises pour créer la nouvelle interface Loopback2. Vous pouvez utiliser le même dictionnaire que vous avez utilisé précédemment dans Postman. Cependant, modifiez le numéro et l'adresse de l'interface. Aussi, sachez que les valeurs booléennes doivent être capitalisées en Python. Par conséquent, assurez-vous que le **T** est mis en majuscule dans la paire clé/valeur pour **"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": {}
    }
}
```

Étape 3: Créez une variable pour envoyer la demande et stocker la réponse JSON.

Utilisez les variables créées à l'étape précédente comme paramètres pour la méthode **requests.put ()**. Cette méthode envoie une requête HTTP PUT à l'API RESTCONF. Affectez le résultat de la requête à une variable nommée **resp**. Cette variable contiendra la réponse JSON de l'API. Si la requête est réussie, le JSON contiendra le modèle de données YANG retourné.

a. Avant d'entrer des instructions, veuillez noter que cette spécification de variable ne doit être que sur une seule ligne de votre script. Entrez les instructions suivantes :

Remarque : Cette spécification de variable doit se trouver sur une ligne dans votre script.

```
resp = requests.put (api_url, data=json.dumps (YangConfig), auth=basicauth, en-têtes
=, verify=false)
```

b. Entrez le code ci-dessous pour gérer la réponse. Si la réponse est l'un des messages de succès HTTP, le premier message sera imprimé. Toute autre valeur de code est considérée comme une erreur. Le code de réponse et le message d'erreur seront imprimés dans le cas où une erreur a été détectée.

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

Le tableau ci-dessous énumère les différents éléments de ces énoncés :

Élément	Explication
`resp`	La variable pour contenir la réponse de l'API.
`requests.put()`	Méthode qui effectue la requête PUT.

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

Élément	Explication
`api_url`	Variable qui contient la chaîne d'adresse URL.
`données`	Données à envoyer au point de terminaison de l'API, qui est formatée en JSON.
`auth`	Variable tuple créée pour contenir les informations d'authentification.
`headers=headers`	Paramètre auquel la variable en-têtes est affectée.
`verify=False`	Paramètre qui désactive la vérification du certificat SSL lors de la demande.
`resp.status_code`	Le code d'état HTTP dans la requête API **PUT** répond.

c. Enregistrez et exécutez le script pour envoyer la requête PUT au CSR1kv. Vous devriez recevoir un message **201 Statut créé**. Si ce n'est pas le cas, vérifiez votre code et la configuration du CSR1kv.

d. Vous pouvez vérifier que l'interface a été créée en entrant **show ip interface brief** sur le 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#
```

Programmes utilisés dans ce TP

Les scripts Python suivants ont été utilisés dans ce TP :

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

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco

```
#=============================================================================
#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''
                }
            ]
        },
        "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
```

© 2020 - aa Cisco et/ou ses filiales. Tous droits réservés. Document public de Cisco