تكوين خدمة MPLS L3VPN على موجه PE باستخدام REST-API (IOS-XE)

المحتويات

المقدمة
المتطلبات الأساسية
_
التكوين
الرسم التخطيطي للشبكة
إجراء التكوين
1. إسترداد معرف الرمز المميز
2. إنشاء VRF
3. نقل الواجهة إلى VRF
4. تعيين عنوان IP للواجهة
5. إنشاء بروتوكول BGP واعيا بعامل VRF
6. تحديد جار BGP ضمن عائلة عناوين VRF
المراجع
المختصرات المستخدمة:

المقدمة

يوضح هذا المستند إستخدام برمجة Python لتوفير MPLS L3VPN على موجه PE (Service Edge) باستخدام REST API. يستخدم هذا المثال موجهات Cisco CSR1000V (IOS-XE) كموجهات PE.

مقدمة من: أنورادا بيريرا

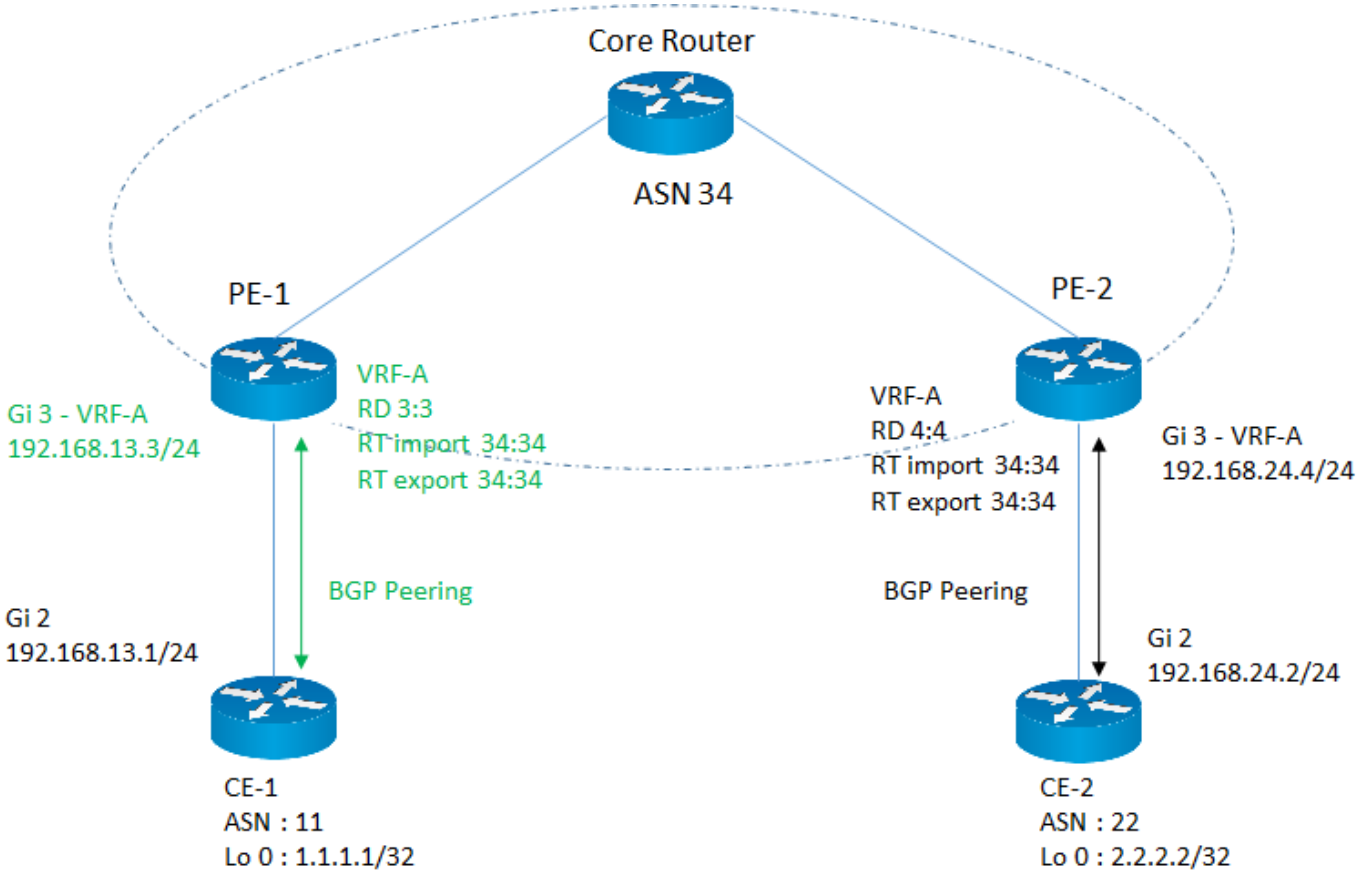
تحرير: كومار سريدهار

المتطلبات الأساسية

- وصول إدارة REST API إلى موجهات CSR1000V (ارجع إلى المراجع في نهاية هذا المستند).
- Python (الإصدار x.2 أو x.3) و"الطلبات" مكتبة Python مثبتة على الكمبيوتر المستخدم لتكوين الموجهات.
- بعض المعرفة الأساسية برمجة بايثون.

التكوين

الرسم التخطيطي للشبكة



في هذا المثال، يكون التركيز على تكوين معلمات خدمة MPLS L3VPN المطلوبة على موجه PE-1، والتي يتم إبرازها بلون وردي.

إجراء التكوين

يتم تقسيم مهمة التكوين إلى عدد من المهام الفرعية ويتم تنفيذ كل مهمة فرعية ضمن وظيفة معرفة من قبل المستخدم. بهذه الطريقة يمكن إعادة إستخدام الوظائف عند الحاجة.

تستخدم جميع الوظائف مكتبة "طلبات" للوصول إلى واجهات برمجة تطبيقات REST على الموجه وتنسيق البيانات هو JSON. في طلبات HTTP، يتم تعيين المعلمة "verify" على "false" لتجاهل التحقق من صحة شهادة SSL.

1. إسترداد معرف الرمز المميز

قبل المتابعة مع أي تكوين على موجه، يلزمك الحصول على معرف مميز صالح تم الحصول عليه من الموجه. تقوم هذه الدالة ببدء طلب HTTP لمصادقة معرف الرمز المميز والحصول عليه حتى يمكنها إستدعاء واجهات API أخرى باستخدام هذا الرمز المميز. تتضمن إستجابة هذا الطلب معرف الرمز المميز.

```
#----------
def GetToken (ip، port، username، كلمة):
طلبات الاستيراد
إستيراد قاعدة 64
"url = https://" + ip +":"+ port + "/api/v1/auth/token-services
الرؤوس = {
'content-type': "application/json"،
التفويض: أساسي + base64.b64encode((username +":"+ كلمة المرور).encode('ascii')).decode('utf-8')،
عنصر التحكم في ذاكرة التخزين المؤقت: "لا توجد ذاكرة تخزين مؤقت"
```

{

الاستجابة = requests.request("POST"، url، رؤوس=رؤوس، verify=خطأ)

إذا كانت response.status_code == 200:

return response.json()['token-id']

غير ذلك:

رجوع "فشل"

#----------

2. إنشاء VRF

ستقوم هذه الوظيفة بإنشاء VRF على موجه PE مع علامة المسار المطلوبة (RD) وأهداف مسار الاستيراد/التصدير (RT)

#----------

def createVRF (ip، port، tokenID، vrfName، rd، importRT، exportRT):

طلبات الاستيراد

url = "https://" + IP + ":"+ المنفذ + "api/v1/vrf/"

الرؤوس = {

'content-type': "application/json"،

'X-auth-token': TokenID،

'control-cache': "لا ذاكرة تخزين مؤقت"

{

البيانات = {

'name': vrfName،

د:

'route-target' : [

}

"action' :"import'،

community': ImportRT'

,{

}

"action' :"export'،

المجتمع: ExportRt

{

[

{

الاستجابة = url، "POST"(requests.request، الرؤوس=false verify، json=data، headers)

إذا كانت response.status_code == 201:

رجوع "ناجح"

غير ذلك:

رجوع "فشل"

#----------

3. نقل الواجهة إلى VRF

ستقوم هذه الوظيفة بنقل واجهة معينة إلى VRF.

#----------

def addInterfacestoVRF (ip، port، tokenID، vrfName، interfaceName، rd، importRT، exportRT):

طلبات الاستيراد

url = "https://" + IP + ":"+ المنفذ + "/api/v1/vrf/" + vrfName

الرؤوس = {

'content-type': "application/json"،

'X-auth-token': TokenID،

'control-cache': "لا ذاكرة تخزين مؤقت"

{

البيانات = {

د:

'forwarding': [interfaceName'،]

] : 'route-target'

}

"action":"import'،

community': ImportRT'

{,

}

"action":"export'،

المجتمع: ExportRt

{

[

{

الاستجابة = url، requests.request("PUT"، الرؤوس=verify=false، json=data، headers)

إذا كانت response.status_code == 204:

رجوع "ناجح"

غير ذلك:

رجوع "فشل"

#----------

4. تعيين عنوان IP للواجهة

ستقوم هذه الوظيفة بتعيين عنوان IP للواجهة.

#----------

def assignInterfaceIP (ip، port، tokenID، interfaceName، interfaceIP، interfaceSubnet):

طلبات الاستيراد

IP = "https://" + url = "api/v1/interfaces/" + interfaceName/" + المنفذ +":"

الرؤوس = {

'content-type': "application/json"،

'X-auth-token': TokenID،

'control-cache': "لا ذاكرة تخزين مؤقت"

{

} = البيانات {

'type': "ethernet"،

'if-name': interfaceName،

'ip-address': interfaceIP،

'subnet-mask': interfaceSubnet

{

(headers، json=data، verify=false=الرؤوس ،requests.request("PUT"، url = الاستجابة

إذا كانت response.status_code == 204:

إرجاع "ناجح"

غير ذلك:

إرجاع "فشل"

#----------

5. إنشاء بروتوكول BGP واعيا بعامل VRF

سيؤدي هذا إلى تمكين IPv4 الخاص بعائلة عناوين VRF.

#----------

def createVrfBGP (ip، port، tokenID، vrfName، ASN):

طلبات الاستيراد

"api/v1/vrf/" + vrfName + "/routing-svc/bgp/" + المنفذ +":" + url = "https://" + IP

} = الرؤوس {

'content-type': "application/json"،

'X-auth-token': TokenID،

'control-cache': "لا ذاكرة تخزين مؤقت"

{

} = البيانات {

routing-protocol-id': ASN'

{

(headers، json=data، verify=false=الرؤوس ،requests.request("POST"، url = الاستجابة

:response.status_code == 201 إذا كانت

رجوع "ناجح"

غير ذلك:

رجوع "فشل"

#----------

6. تحديد جار BGP ضمن عائلة عناوين VRF

ستحدد هذه الوظيفة جار BGP ضمن IPv4 لعائلة عنوان VRF.

#----------

:(def defineVrfBGPNgbor (ip، port، tokenID، vrfName، ASN، neighborsIP، remoteAS

طلبات الاستيراد

API/v1/vrf/" + vrfName + "/routing-svc/bgp/" + ASN/" + المنفذ +":" + url = "https://" + IP "+"/neighbors

الرؤوس = {

'content-type': "application/json"،

'X-auth-token': TokenID'،

'control-cache': "لا ذاكرة تخزين مؤقت"

{

البيانات = {

'routing-protocol-id': ASN'،

'address': الجيران IP،

'remote-as': remoteAS'

{

(headers، json=data، verify=false=الرؤوس ،requests.request("POST"، url = الاستجابة

:response.status_code == 201 إذا كانت

رجوع "ناجح"

غير ذلك:

رجوع "فشل"

#----------

وصف معلمات الإدخال وقيمها

" ip = "10.0.0.1 # عنوان IP للموجه

المنفذ = "55443" # منفذ REST API على الموجه

username = "cisco" إلى تسجيل الدخول.يجب تكوين هذا مع مستوى الامتياز 15.

كلمة السر = "cisco" # كلمة المرور المقترنة باسم المستخدم

TokenID = <القيمة التي تم إرجاعها> # معرف Token الذي تم الحصول عليه من الموجه باستخدام وظيفة GetToken

"VRF-A = "vrfName # اسم ال VRF

"3:3 = "RD # مميزات المسار ل VRF

"34:34 = "ImportRT # إستيراد هدف المسار

التصدير"34:34 = "RT # هدف مسار التصدير

اسم الواجهة = "GigabitEthernet3" # اسم واجهة واجهة حافة العميل (CE)

"192.168.13.3 = "interfaceIP # عنوان IP لواجهة واجهة CE الموجهة

"255.255.255.0 = "InterfaceSubnet# شبكة فرعية للواجهة الموجهة CE

ASN = "34" # BGP كعدد من موجه PE

مجاور BGP Peering IP # "192.168.13.1" = IP من موجه CE

"11 = "RemoteAS # كعدد من موجه CE

وفي جميع الوظائف المذكورة أعلاه، تم إستدعاء واجهات برمجة تطبيقات مخصصة لكل عملية تكوين. يوضح المثال التالي كيفية تمرير واجهة سطر الأوامر IOS-XE، بشكل عام، في متن واجهة برمجة تطبيقات REST. يمكن إستخدام هذا كحل بديل لأتمتة حالة عدم توفر واجهة برمجة تطبيقات معينة. في الدالات المذكورة أعلاه، يتم تعيين "نوع المحتوى" على "التطبيق/json"، ولكن في المثال التالي، يتم تعيين "نوع المحتوى" على "نص/عادي" حيث إنه يقوم بتحليل إدخال واجهة سطر الأوامر (CLI) القياسي.

يحدد هذا المثال وصف الواجهة للواجهة GigabitEthernet3. يمكن تخصيص التكوين بتغيير المعلمة "cliInput".

#----------

def passCLIInput (ip، port، tokenID):

```
طلبات الاستيراد

"url = https://" + ip + ":+ port + "/api/v1/global/running-config

الرؤوس = {

'content-type': "text/plain"،

'X-auth-token': TokenID،

'control-cache' : "لا ذاكرة تخزين مؤقت"

{

line1 = "واجهة GigabitEthernet 3"

السطر 2 = "واجهة واجهة واجهة واجهة العميل التي تدعم الوصف"

cliInput = line1 + "\r\n" + line2

الاستجابة = requests.request("PUT"، url، رؤوس=رؤوس، verify=false ،data=cliInput )

طباعة(response.text)

إذا كانت response.status_code == 204:

رجوع "منتصر"

غير ذلك:

رجوع "فشل"

#----------
```

المراجع

- دليل تكوين برنامج موجه خدمات السحابة Cisco CSR 1000V Series Cloud Services Router Software

https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/configuration/b_CSR1000v_Configuration_Guide /b_CSR1000v_Configuration_Guide_chapter_01101.html

- دليل مرجع إدارة واجهة برمجة التطبيقات IOS XE REST من Cisco

https://www.cisco.com/c/en/us/td/docs/routers/csr1000/software/restapi/restapi.html

المختصرات المستخدمة:

MPLS - تحويل التسمية متعدد البروتوكولات

L3 - الطبقة 3

VPN - الشبكة الخاصة الظاهرية

VRF - إعادة توجيه المسار الظاهري

BGP - بروتوكول العبارة الحدودية

الراحة - نقل الحالة التمثيلية

واجهة برنامج التطبيق - API

تدوين كائن Java Script

HTTP - بروتوكول نقل النص التشعبي

حول هذه الترجمة
ترجمت Cisco هذا المستند باستخدام مجموعة من التقنيات الآلية والبشرية لتقديم محتوى دعم للمستخدمين في جميع أنحاء العالم بلغتهم الخاصة. يُرجى ملاحظة أن أفضل ترجمة آلية لن تكون دقيقة كما هو الحال مع الترجمة الاحترافية التي يقدمها مترجم محترف. تخلي Cisco Systems مسئوليتها عن دقة هذه الترجمات وتُوصي بالرجوع دائمًا إلى المستند الإنجليزي الأصلي (الرابط متوفر).