

ご利用のコンピュータを設定する方法

このラボの作業を行うには、事前設定済みの dCloud ラボを使用するか、自身のコンピュータをセットアップしてください。詳細については、[イベントの準備](#) [英語] モジュールと[ラボの設定](#) [英語] モジュールを確認してください。

APIC-EM REST API

APIC-EM のノースバウンド (NB) REST ベースの API により、ネットワーク エンジニアとオペレータは、カスタム アプリケーションを使用してコントローラと接続し、ネットワーク状況の変化や関連するビジネス チャンスに迅速に対応できるようになります。

APIC-EM サウスバウンド インターフェイスはネットワークの制御プレーンに接続するもので、直接のアクセスはできません。これは、ノースバウンド (NB) REST API を通じて特定されたネットワーク デバイスでコマンドを実行するために使用されます。したがって、APIC-EM と通信して動的な SDN 機能をネットワークに直接追加するように、アプリケーションを設計できます。

このラボでは、Python プログラミング言語によって Cisco APIC-EM ノースバウンド REST API を使用方法を示します。

目標

所要時間:35 分

- Postman REST クライアントを使用して Python コードを生成する方法を学習する
- Python を使用してカスタム スクリプトを記述する

前提条件

このモジュールでは、REST API コールを発信する Google Chrome ブラウザのアプリケーションである [Postman](#) を使用します。

また、プログラミング言語として [Python3](#) を使用します。

背景:このラボを開始する前に、ラーニング ラボのモジュールを完了しておくことを強くお勧めします。

- [ラボの設定](#) [英語]
- [REST API および Python](#) [英語]

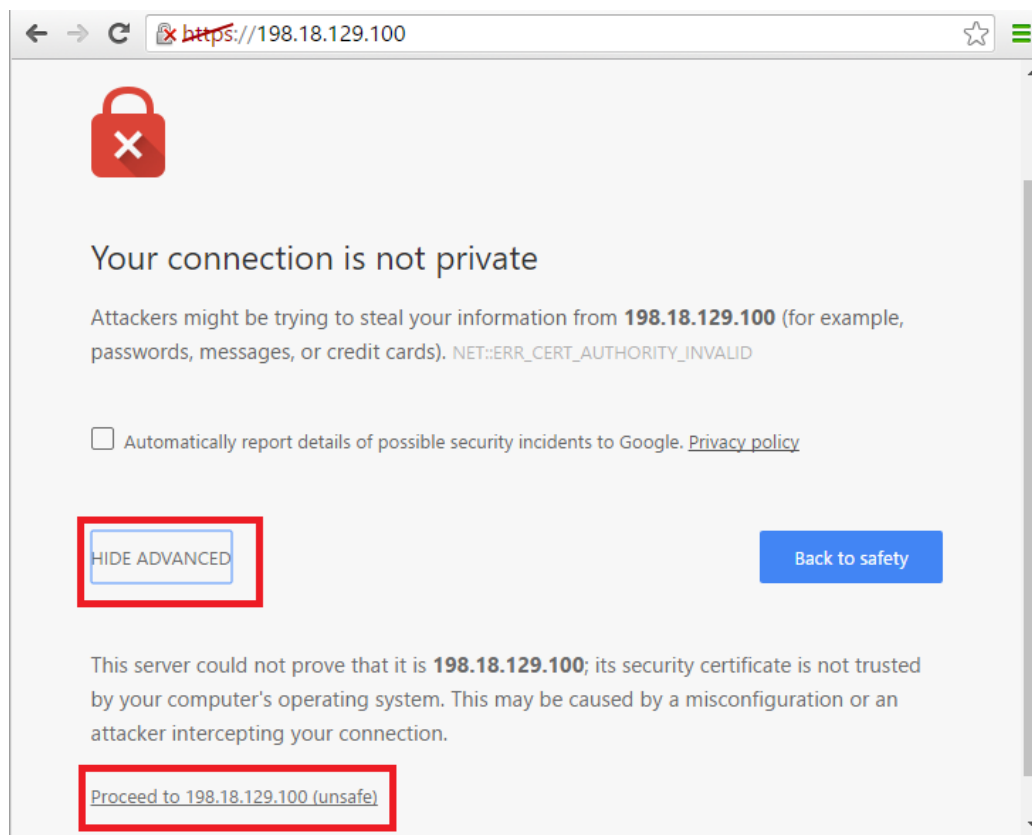
詳細については、「自身のコンピュータを設定する方法」の上のボックスをクリックしてください。

APIC-EM コントローラへのアクセス

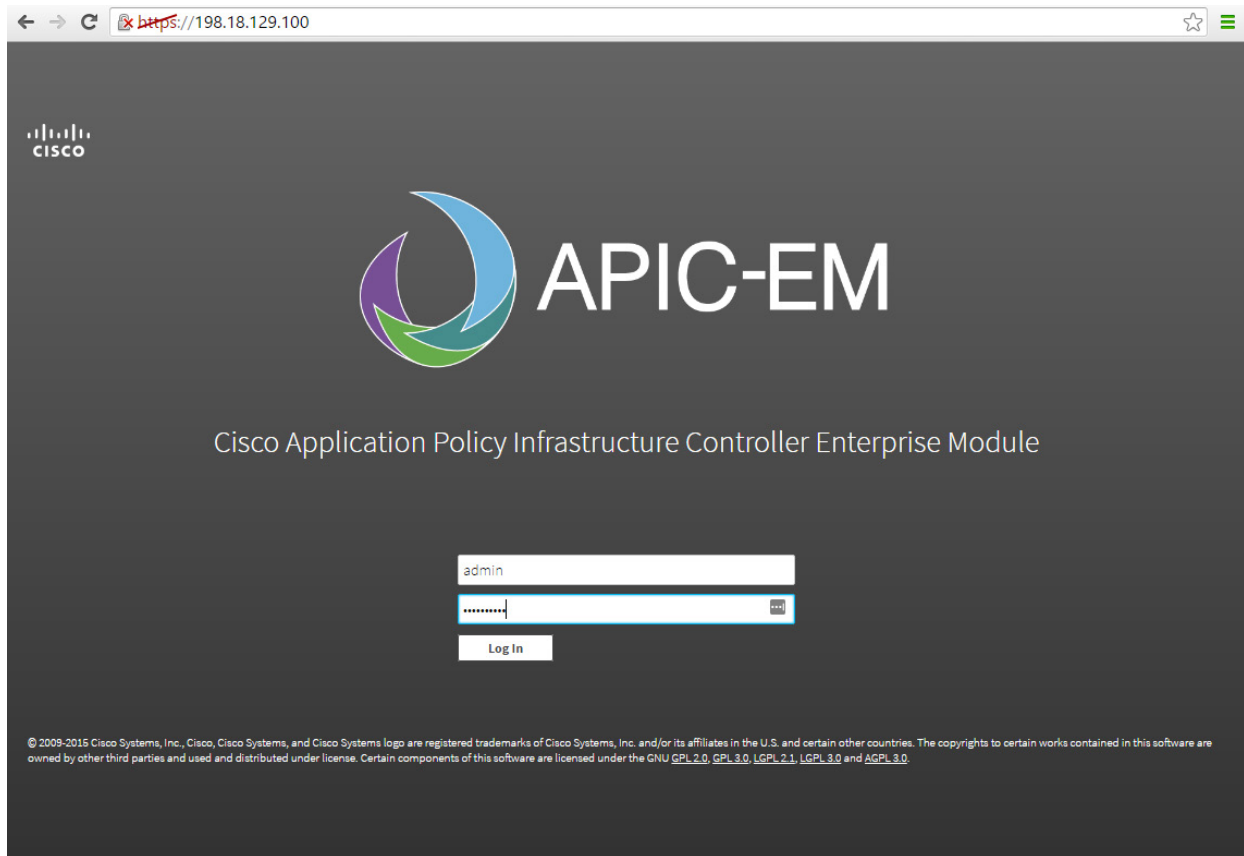
- これらのサンプル コードを実行するためには、APIC-EM コントローラにアクセスできる必要があります。
- APIC-EM コントローラをまだ導入していない場合は、Cisco dCloud から提供された APIC-EM コントローラを使用できます。
 - APIC-EM コントローラを利用できる dCloud に接続する方法については、[ラボのセットアップ](#) [英語] モジュールを参照してください。
 - 接続すると、IP アドレス <https://198.18.129.100/> を使用して APIC-EM コントローラにアクセスできるようになります。コントローラのログイン クレデンシャルは、ユーザ名:admin、パスワード:C1sco12345 です。

ステップ 1: APIC-EM に接続する

Web ブラウザを開き、<https://198.18.129.100/> の Web ページに移動します。SSL 証明書の警告ページが表示されます。続行するには、[詳細 (Advanced)] ボタンをクリックして、[198.18.129.100 に進む (Proceed to 198.18.129.100)] リンクをクリックします。

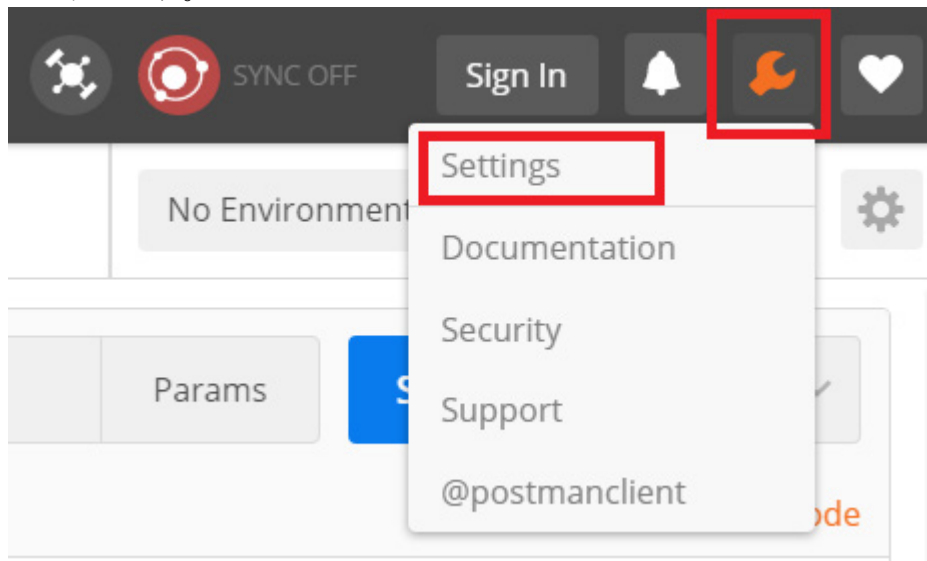


ログイン ページで、ユーザ名 admin、パスワード C1sco12345 を使用して認証します。

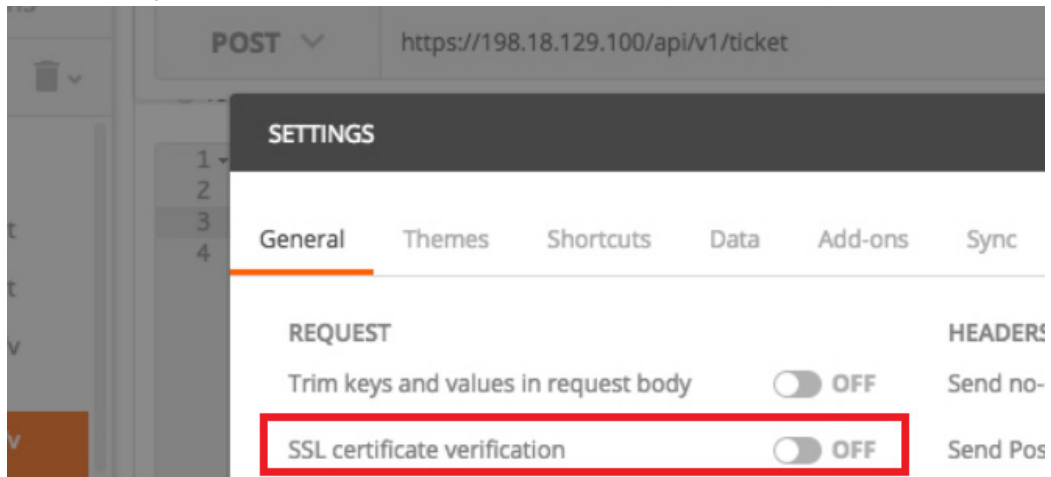


Mac で Postman を使用する場合は、SSL 検証がオフになっていることを確認してください。

1. Postman でスパナ アイコンをクリックして、ドロップダウン メニューから [設定 (Settings)] を選択します。



2. [全般 (General)] タブのポップアップ ウィンドウで、SSL 証明書がオフになっていることを確認します。このオプションが表示されない場合は、そのままウィンドウを閉じてください。



次に、[Postman](#) アプリケーションを使用して Python コードを生成します。

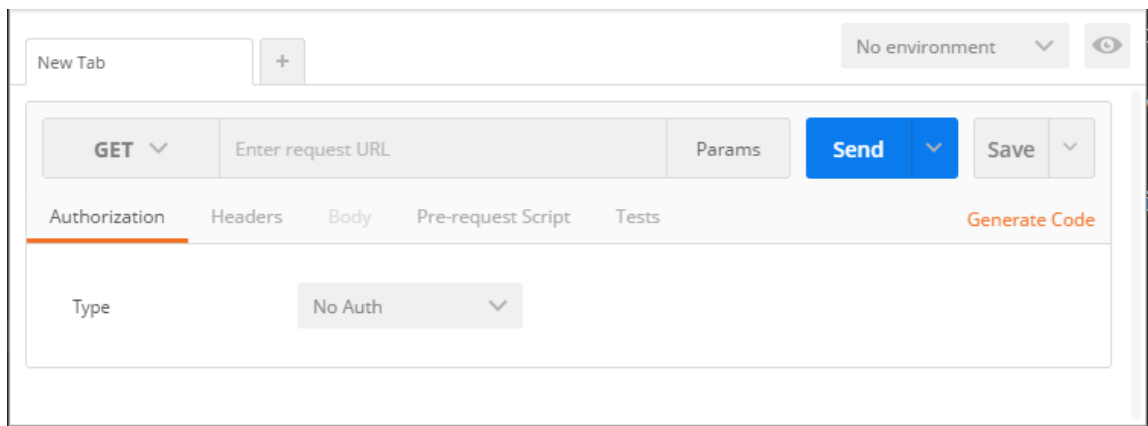
ステップ 2: Postman を使用してコードを生成する

ラーニング ラボの [REST API および Python](#) [英語] モジュールで、Postman アプリケーションを使用して API リクエストを行う方法を説明しました。ここでは、Postman を使用して、各種のソフトウェア プログラミング 言語でコードを生成する方法を示します。このステップでは、アプリケーションの Python 言語 オプションを使用します。

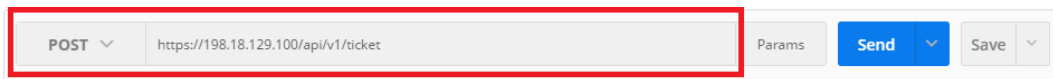
ここでは、Postman アプリケーションと Python がワークステーションにインストール済みで、使用方法も理解していることを前提としています。システムをセットアップしていない場合は、このページの最初の「[自分のコンピュータを設定する方法](#)」セクションを参照してください。

コードを生成するには、Postman で関連するフィールドに入力する必要があります。それでは始めましょう。

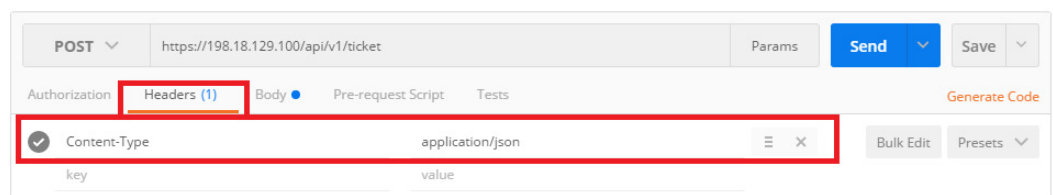
1. ワークステーションで Postman アプリケーションを開きます。



2. 最初に、APIC-EM から認証トークンを取得して、API コールを可能にします。それには、メソッドを POST に設定し、URL を `https://198.18.129.100/api/v1/ticket` に設定します。

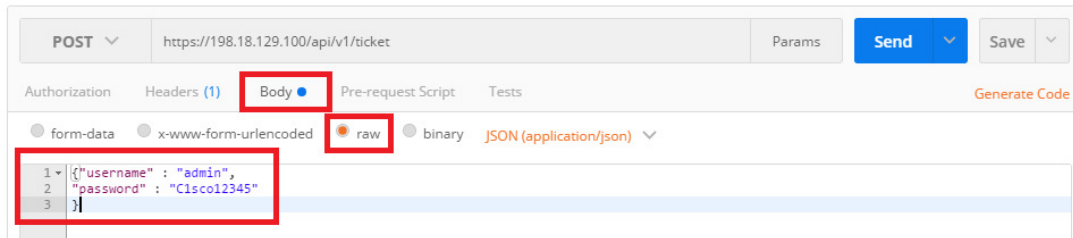


3. 次に、リクエスト ヘッダー情報を定義する必要があります。ヘッダー タブをクリックして、1 つのキー値ペアを入力します。
 - [Content-Type] および [application/json]

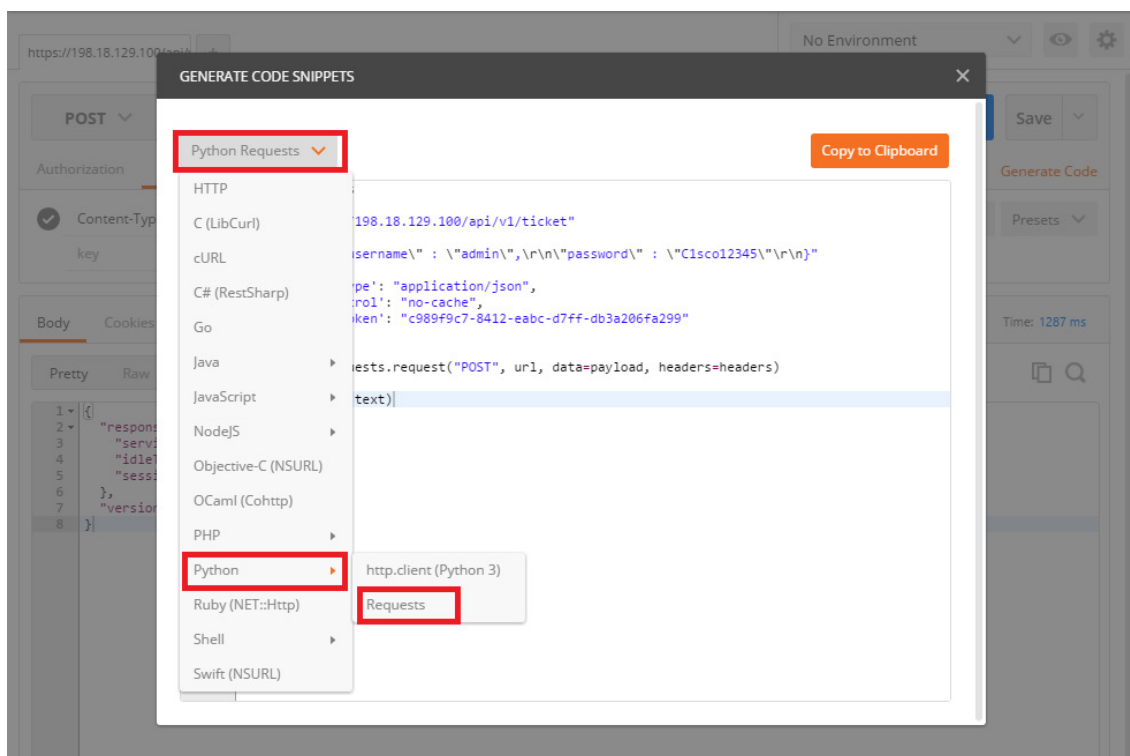


4. 最後に、認証トークンを取得するために、ユーザ名とパスワードを入力します。[ボディ (Body)] タブをクリックして、[raw] オプションを選択します。下のウィンドウに次の情報を貼り付けます(この情報は POST のデータとして送信されます)。

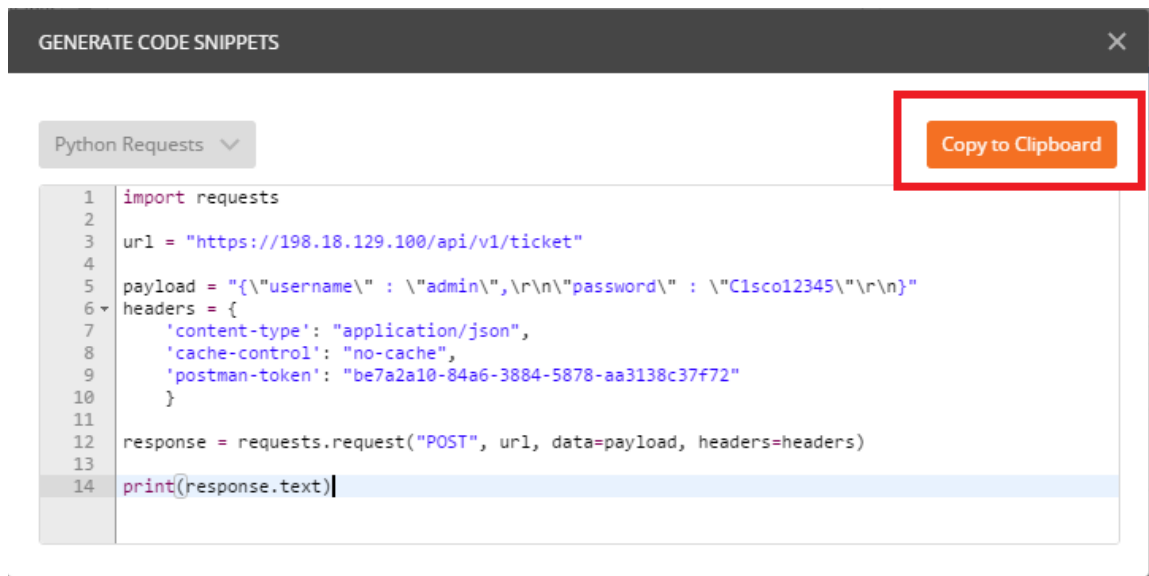
```
5. {  
6.   "username" : "admin",  
7.   "password" : "Cisco12345"  
8. }
```



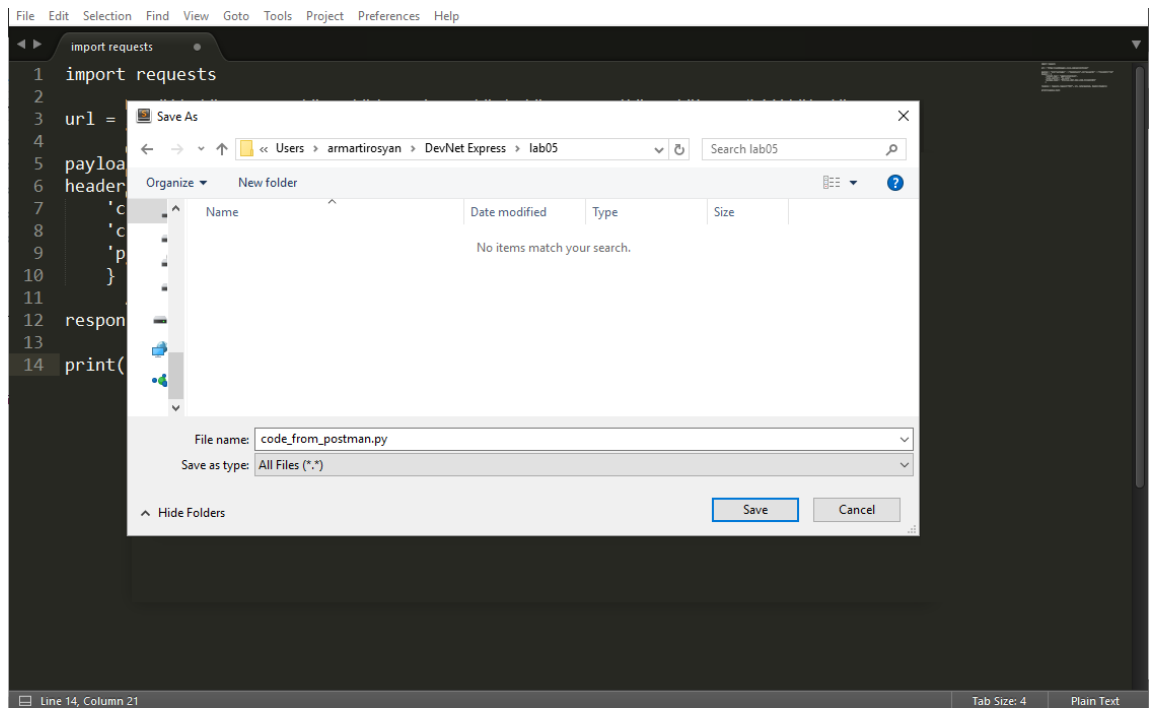
9. [送信 (Send)] ボタンをクリックして、すべてが正常に機能することと、入力した情報が正しいことを確認します。その結果、有効なトークンと 200/OK が得られます。
10. これで、Postman を使用してコードを生成する準備ができました。[保存 (Save)] ボタンの下に、[コードの生成 (Generate Code)] リンクがあります。リンクをクリックします。[コード スニペットの生成 (Generate Code Snippets)] ウィンドウで、[Python] -> [リクエスト (Requests)] の順に選択します。



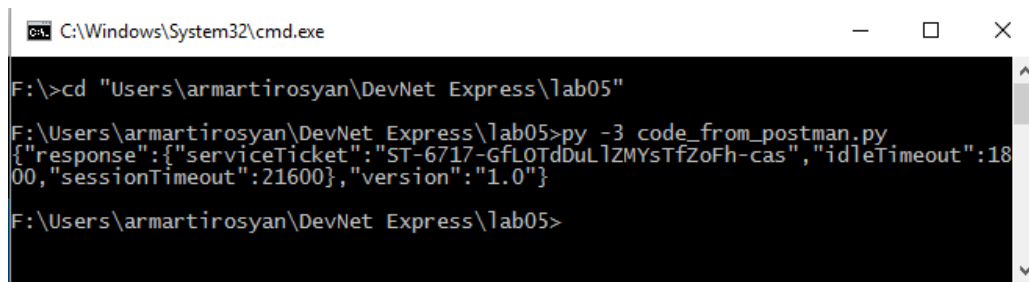
11. requests モジュールによってエントリが Python コードに変換され、API コールが可能になります。[クリップボードにコピー (Copy to Clipboard)] ボタンをクリックしてコードをコピーします。



12. クリップボードの内容を任意のテキスト エディタに貼り付け、作業ディレクトリに Python ファイルとして保存します。



13. 正しく処理できるか試してみてください。コマンドライン ターミナルを開き、作業ディレクトリに移動します。py -3 <FILE-NAME.py> コマンドを実行します。APIC-EM コントローラからの応答が確認できるはずです。



```
C:\Windows\System32\cmd.exe
F:\>cd "Users\armartirosyan\DevNet Express\lab05"
F:\Users\armartirosyan\DevNet Express\lab05>py -3 code_from_postman.py
{"response":{"serviceTicket":"ST-6717-GfLOTdDuLlZMYsTfZoFh-cas","idleTimeout":1800,"sessionTimeout":21600},"version":"1.0"}
F:\Users\armartirosyan\DevNet Express\lab05>
```

14. SSL エラーが表示された場合は、リクエスト関数コールに "Verify=False" を追加し、スクリプトを再度実行します。

```
import requests

url = "https://198.18.129.100/api/v1/ticket"

payload = "{\"username\":\"admin\", \"password\":\"Cisco12345\"}"
headers = {
    'content-type': "application/json",
    'cache-control': "no-cache",
    'postman-token': "8f0fbcd6-46c8-1fab-86d1-462f86068deb"
}

response = requests.request("POST", url, data=payload, headers=headers, verify=False)

print(response.text)
```

以上で、Postman を使用してコードを生成する方法を学習しました。このラボの次のセクションでは、自身でコードを記述する方法について示します。

ステップ 3: Python スクリプトを記述する

ここでは自身でコードを記述してみましょう。手順に従ってコードを記述すれば、API リクエストが APIC-EM に送信され、認証トークンが取得されます。

- Python を使用して API コールを行うには、requests と json の 2 つのモジュールをインポートする必要があります。

```
#Import necessary modules
import requests
import json
```

注:JSON モジュールの機能は requests モジュールにすでに含まれているため、技術的には JSON モジュールは必須ではありません。たとえば、JSON データは、`data=json.dumps(variable)` ではなく `json=variable` パラメータを使用して `requests.post` を呼び出すことで、POST メソッドに渡すことができます。ただし、受講者が json モジュールと `loads()` および `dumps()` メソッドの機能を理解することには意義があります。

- モジュールに含まれている必要な機能は、すべて呼び出すことができません。オプションで、SSL 証明書に関連する警告メッセージをオフにすることができます。それには次のコードを使用します。

```
#Import necessary modules
import requests
import json

#Disable warnings
requests.packages.urllib3.disable_warnings()
```

- 次に、後でコード内で使用する変数を定義します。

```
#Import necessary modules
import requests
import json

#Disable warnings
requests.packages.urllib3.disable_warnings()

# Variables

apic_em_ip = "https://198.18.129.100/api/v1"
api_call = "/ticket"
```

- 次に、ペイロード、ヘッダー、パラメータ情報を定義します。

```
#Import necessary modules
import requests
import json
```

```

#Disable warnings
requests.packages.urllib3.disable_warnings()

# Variables

apic_em_ip = "https://198.18.129.100/api/v1"
api_call = "/ticket"

#Payload contains authentication information
payload = {"username":"admin","password":"C1sco12345"}

# Header information
headers = {"content-type" : "application/json"}

```

- この呼び出しの結果を、response 変数に割り当てます。

```

#Import necessary modules
import requests
import json

#Disable warnings
requests.packages.urllib3.disable_warnings()

# Variables
apic_em_ip = "https://198.18.129.100/api/v1"
api_call = "/ticket"

#Payload contains authentication information
payload = {"username":"admin","password":"C1sco12345"}

# Header information
headers = {"content-type" : "application/json"}

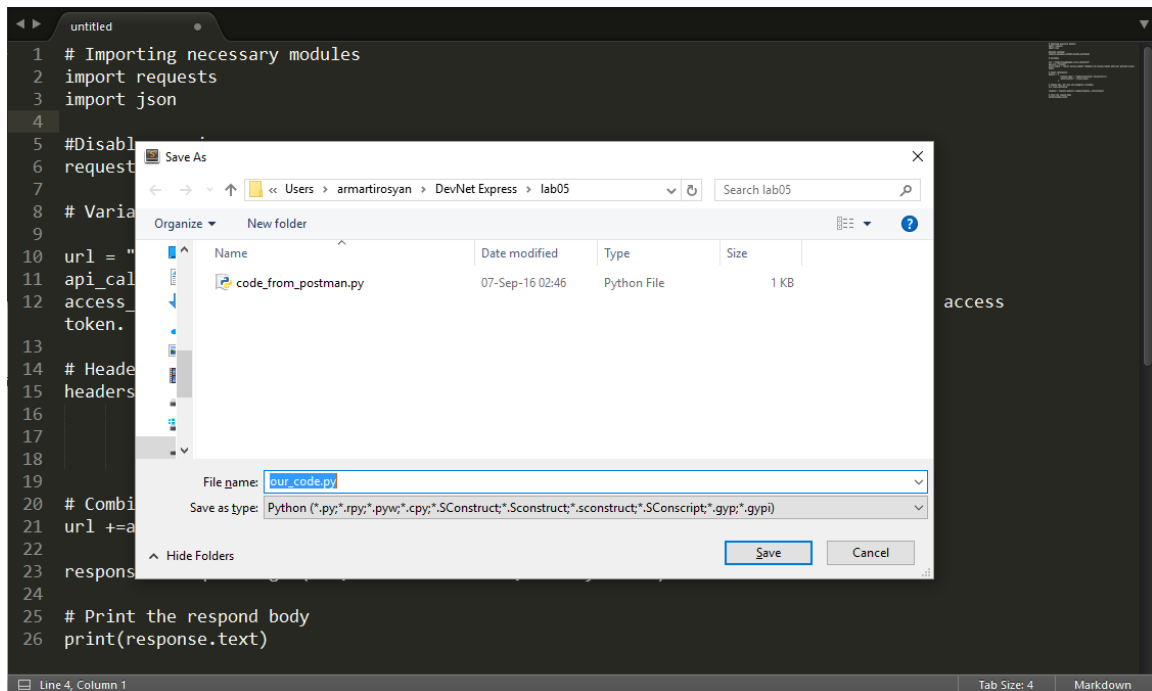
# Combine apic_em_ip and api_call variables into one variable call url
url = apic_em_ip + api_call

response = requests.post(url, data=json.dumps(payload), headers=headers,
verify=False)

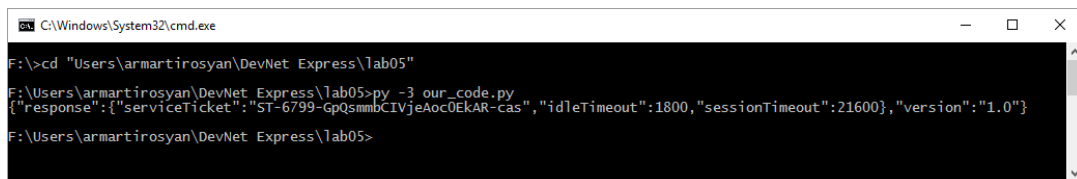
# Print the respond body
print(response.text)

```

- これでコードを使用できるようになりました。コードは、Python ファイルとして作業ディレクトリにコピーして保存できます。



- どのように機能するかを確認するには、コマンドライン インターフェイスを開き、作業ディレクトリに移動します。次に `py -3 our_code.py` コマンドを実行します。すべて正しく記述されていれば、端末画面に出力が表示されます。



ご覧のように、出力は読みやすいものではありません。コードを変更して、必要な情報だけが出力されるようにしてみましょう。次のようにします。

1. `response = requests.post(url, data=json.dumps(payload), headers=headers, verify=False)` 行の最後に、`.json()` を追加します。
 2. `print(response.text)` 行を `print("Authentication Token: " + response["response"]["serviceTicket"])` に置き換えます。
 3. これで、`serviceTicket` キーに保存されている値のみが出力されるようになります。
- コードは最終的に次のようになります。

```

#Import necessary modules
import requests
import json

#Disable warnings
requests.packages.urllib3.disable_warnings()

```

```
# Variables
apic_em_ip = "https://198.18.129.100/api/v1"
api_call = "/ticket"

#Payload contains authentication information
payload = {"username":"admin","password":"C1sco12345"}

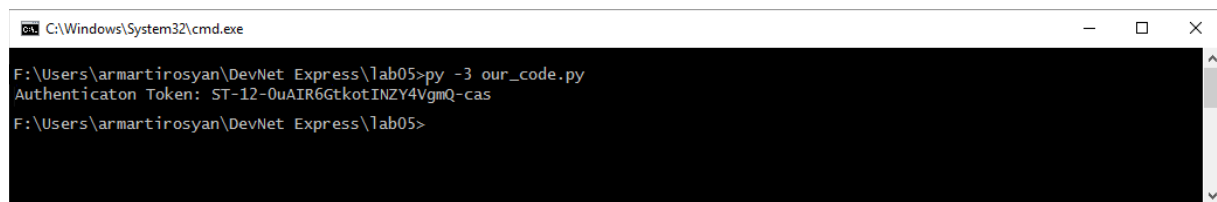
# Header information
headers = {"content-type" : "application/json"}

# Combine apic_em_ip and api_call variables into one variable call url
url = apic_em_ip + api_call

response = requests.post(url, data=json.dumps(payload), headers=headers,
verify=False).json()

# Print the authentication token from respond body
print("Authenticaton Token: " + response["response"]["serviceTicket"])
```

コードを変更して再度実行します。出力が簡潔で読みやすくなります。



```
C:\Windows\System32\cmd.exe
F:\Users\armartirosyan\DevNet Express\lab05>py -3 our_code.py
Authenticaton Token: ST-12-0uAIR6GtkotINZY4VgmQ-cas
F:\Users\armartirosyan\DevNet Express\lab05>
```

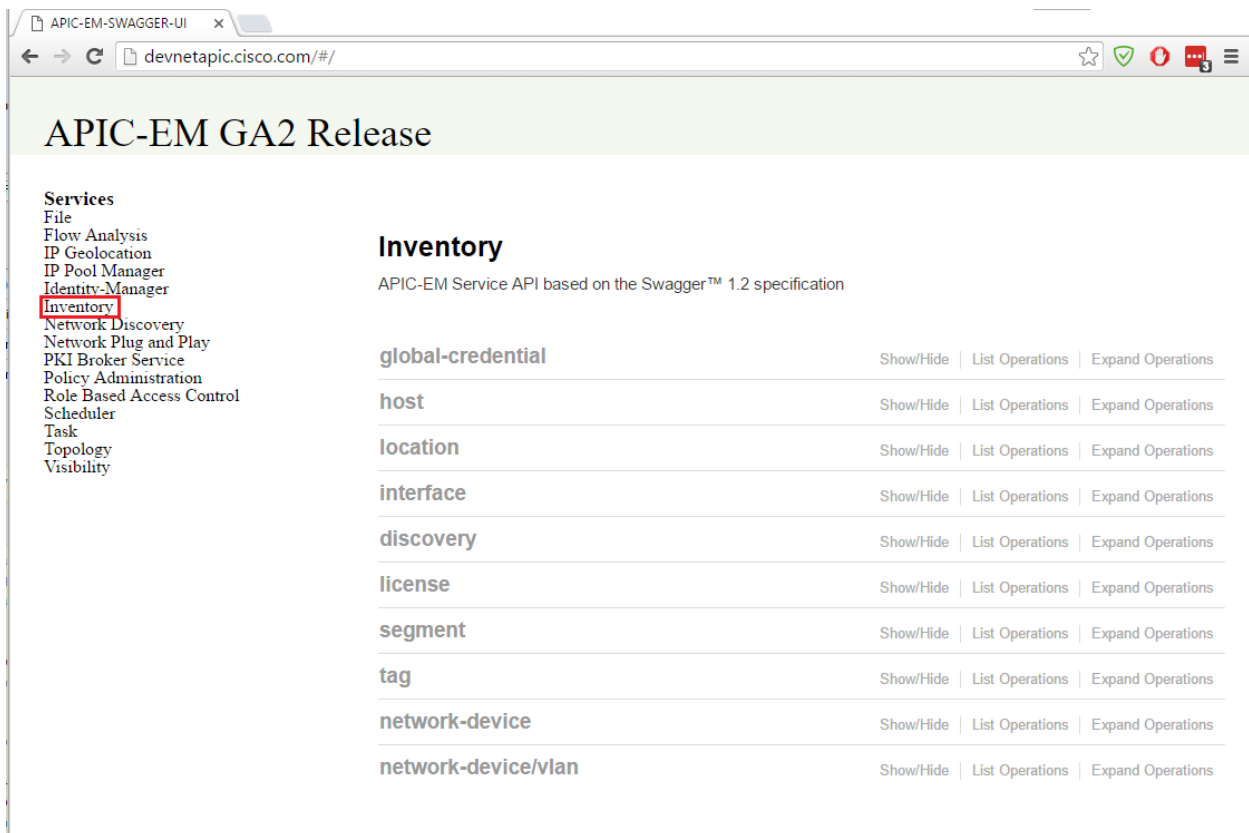
おめでとうございます。REST API コールを行い、APIC-EM から情報を取得する Python コードを記述しました。次のステップでは、ネットワーク デバイスを APIC-EM のインベントリリストから選択し、そのデバイスで実行されている設定を取得した上でファイルに保存するアプリケーションを作成します。

ステップ 4: Python アプリケーション

このステップでは、ネットワーク デバイスから設定を取得してファイルに保存するアプリケーションを作成します。デバイスは APIC-EM のインベントリリストから選択します。

API コールを正しく行うには、どのメソッドを使用するか、またどのような情報をリクエストに含めるかを理解する必要があります。それには、APIC-EM の API ドキュメント (<http://devnetapic.cisco.com/> [英語]) を参照してください。

ここで作成するアプリケーションでは、[インベントリ(Inventory)] セクションでグループ化されている API コールを主に使用します。ここで内容を確認しておいてください。



The screenshot shows the APIC-EM GA2 Release Swagger UI in a web browser. The browser tab is labeled 'APIC-EM-SWAGGER-UI' and the address bar shows 'devnetapic.cisco.com/#/'. The page title is 'APIC-EM GA2 Release'. On the left, there is a sidebar menu with the following items: Services, File, Flow Analysis, IP Geolocation, IP Pool Manager, Identity-Manager, **Inventory** (highlighted with a red box), Network Discovery, Network Plug and Play, PKI Broker Service, Policy Administration, Role Based Access Control, Scheduler, Task, Topology, and Visibility. The main content area is titled 'Inventory' and includes the subtitle 'APIC-EM Service API based on the Swagger™ 1.2 specification'. Below this, there is a table listing various API endpoints with their corresponding actions.

Endpoint	Show/Hide	List Operations	Expand Operations
global-credential			
host			
location			
interface			
discovery			
license			
segment			
tag			
network-device			
network-device/vlan			

アプリケーションを作成する準備ができました。

- 最初に、前のステップのコードをコピーして、関数に変換します。最初の関数では、APIC-EM の URL アドレスを 1 つの引数として使用することで、APIC-EM コントローラから取得した認証トークンが返されます。返されたトークンは、他の関数で使用され、必要な API コールが行われます。
- def get_token(url):
-
- api_call = "/ticket"

-
- #Payload contains authentication information
- payload = { "username": "admin", "password": "C1sco12345" }
-
- # Header information
- headers = {"content-type" : "application/json"}
-
- # Combine URL, API call variables
- url +=api_call
-
- response = requests.post(url, data=json.dumps(payload), headers=headers, verify=False).json()
-
- # Return authentication token from respond body
- return response["response"]["serviceTicket"]
- 次に、2つの引数、認証トークン、APIC-EMのURLアドレスを使用して、APIC-EMから(GET)インベントリ情報を取得する新しい関数を作成します。取得した情報は、定義した基準によってフィルタリングされます。最初の照合では、一致したデバイスのIDが返されます。
- def get_device_id(token, url):
-
- #Define API Call
- api_call = "/network-device"
-
- #Header information
- headers = {"X-AUTH-TOKEN" : token}
-
- # Combine URL, API call variables
- url +=api_call
-
- response = requests.get(url, headers=headers, verify=False).json()
-
- #Iterate over the response and find first device with access role.
- #Return ID number of the first device matching the the `if` statement
- for item in response['response']:
- if item['role'] == 'ACCESS':
- return item['id']
- 最後に、前の2つの関数によって返された値を使用して、特定のデバイスから設定を取得し、ファイルに書き込みます。そのために、3つの引数(認証トークン、APIC-EMのURLアドレス、ネットワークデバイスのID)を取る新しい関数を作成し、APIC-EMにクエリを行ってデバイスの設定を取得する必要があります。
- def get_config(token, url, id):
-
- #Define API Call.To get specific device's configuration
- #we will need to add device's ID in the API call
- api_call = "/network-device/" + id + "/config"
-
- #Header information

- headers = {"X-AUTH-TOKEN" : token}
-
- # Combine URL, API call variables
- url +=api_call
-
- response = requests.get(url, headers=headers, verify=False).json()
-
- #Create a file in present working directory
- file = open('access_host_1.txt', 'w')
-
- #Write response body to the file
- file.write(response['response'])
-
- #Close the file when writing is complete
- file.close()
- 以上で、すべての関数が作成されたので、それらをまとめます。コードは次のようになります。

```
#import modules
import requests
import json

#Disable warnings
requests.packages.urllib3.disable_warnings()

# Variables
apic_em_ip = "https://198.18.129.100/api/v1"

def get_token(url):

    #Define API Call
    api_call = "/ticket"

    #Payload contains authentication information
    payload = { "username": "admin", "password": "C1sco12345" }

    #Header information
    headers = {"content-type" : "application/json"}

    #Combine URL, API call and parameters variables
    url +=api_call

    response = requests.post(url, data=json.dumps(payload), headers=headers,
verify=False).json()

    # Return authentication token from respond body
    return response["response"]["serviceTicket"]

def get_device_id(token,url):

    #Define API Call
    api_call = "/network-device"
```

```

#Header information
headers = {"X-AUTH-TOKEN" : token}

# Combine URL, API call and parameters variables
url +=api_call

response = requests.get(url, headers=headers, verify=False).json()

#Iterate over the response and find first device with access role.
#Return ID number of the first device matching the criteria
for item in response['response']:
    if item['role'] == 'ACCESS':
        return item['id']

def get_config(token, url, id):

    #Define API Call.To get specific device's configuration
    #we will need to add device's ID in the API call
    api_call = "/network-device/" + id + "/config"

    #Header information
    headers = {"X-AUTH-TOKEN" : token}

    # Combine URL, API call variables
    url +=api_call

    response = requests.get(url, headers=headers, verify=False).json()

    #Create a file in present working directory
    file = open('access_host_1.txt', 'w')

    #Write response body to the file
    file.write(response['response'])

    #Close the file when writing is complete
    file.close()

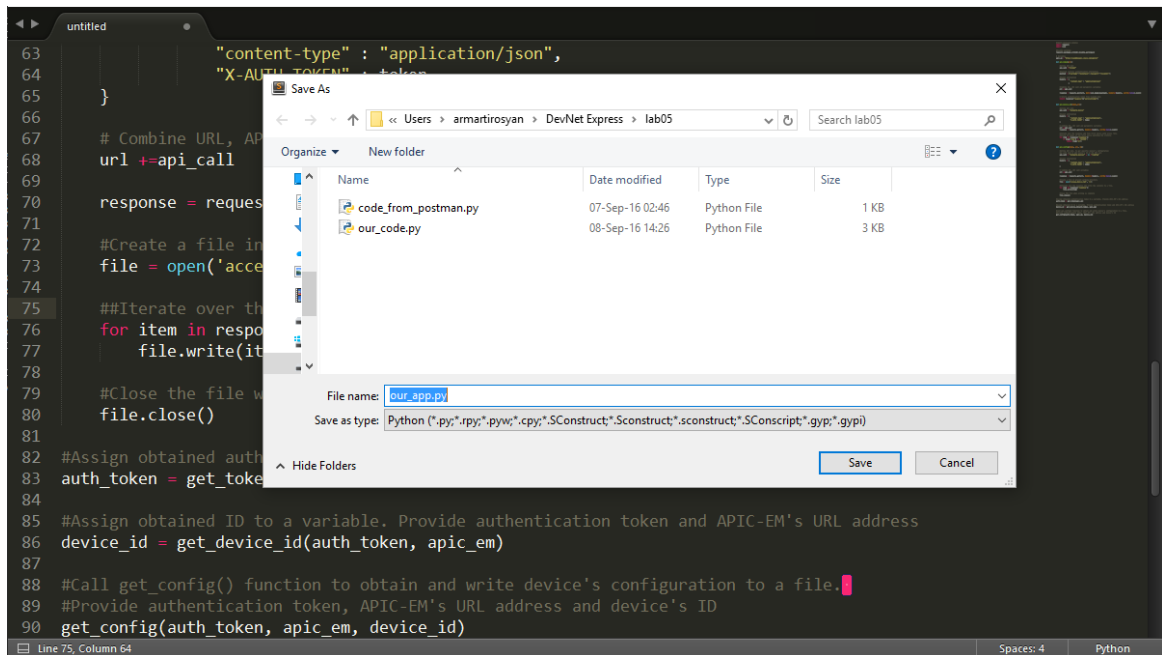
#Assign obtained authentication token to a variable.Provide APIC-EM's URL
address
auth_token = get_token(apic_em_ip)

#Assign obtained ID to a variable.Provide authentication token and APIC-EM's
URL address
device_id = get_device_id(auth_token, apic_em_ip)

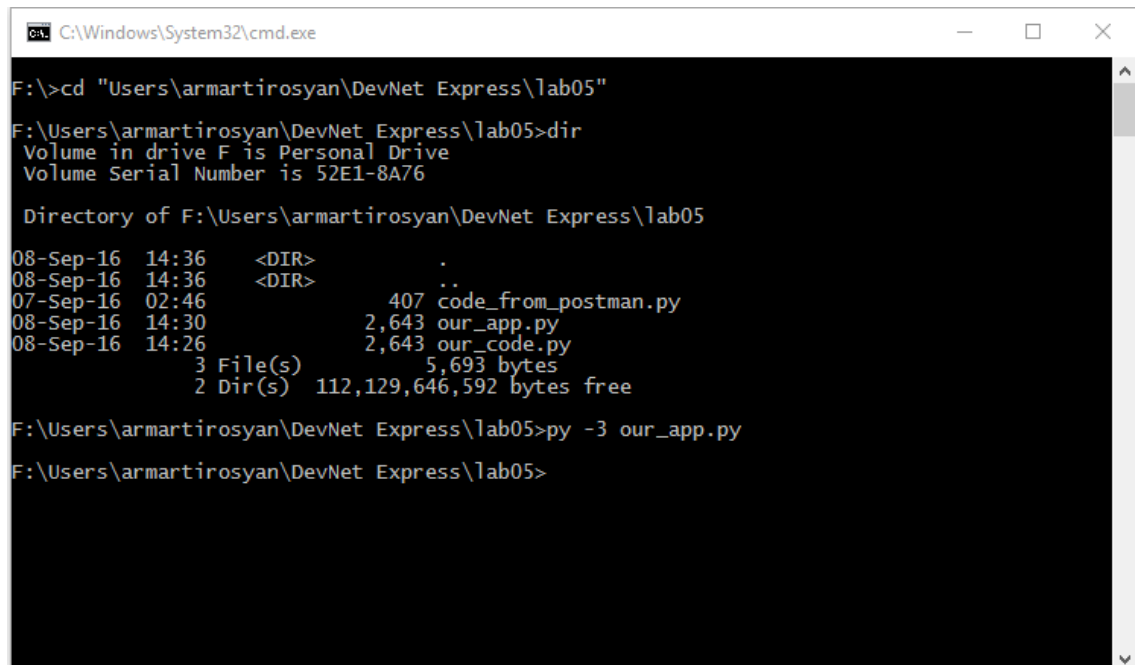
#Call get_config() function to obtain and write device's configuration to a
file.
#Provide authentication token, APIC-EM's URL address and device's ID
get_config(auth_token, apic_em_ip, device_id)

```

- 上記のコードをコピーして、Python ファイルとして保存します。



- ターミナル ウィンドウを開き、Python コードが保存されているディレクトリに移動します。
`py -3 our_app.py` または `python our_app.py` コマンドを実行して、アプリケーションを起動します。すべてが正しく実行されると、端末画面に新しいプロンプトが表示されます。これは、コードの実行中にエラーが発生しなかったことを示します。



- 次にターミナル ウィンドウでフォルダの内容を再度確認します。access_host_1.txt という名前の新しいファイルができています。このファイルはアプリケーションによって作成されたものです。

```
C:\Windows\System32\cmd.exe
F:\>cd "Users\armartirosyan\DevNet Express\lab05"
F:\Users\armartirosyan\DevNet Express\lab05>dir
Volume in drive F is Personal Drive
Volume Serial Number is 52E1-8A76

Directory of F:\Users\armartirosyan\DevNet Express\lab05

08-Sep-16 14:36    <DIR>        .
08-Sep-16 14:36    <DIR>        ..
07-Sep-16 02:46             407 code_from_postman.py
08-Sep-16 14:30           2,643 our_app.py
08-Sep-16 14:26           2,643 our_code.py
                3 File(s)      5,693 bytes
                2 Dir(s)  112,129,646,592 bytes free

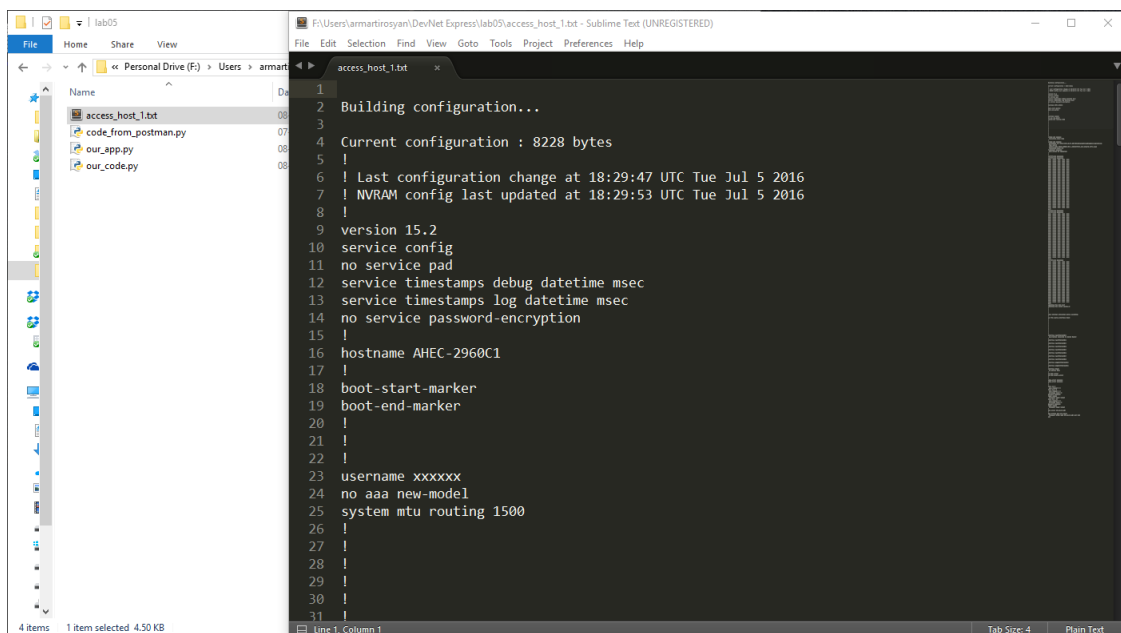
F:\Users\armartirosyan\DevNet Express\lab05>py -3 our_app.py
F:\Users\armartirosyan\DevNet Express\lab05>dir
Volume in drive F is Personal Drive
Volume Serial Number is 52E1-8A76

Directory of F:\Users\armartirosyan\DevNet Express\lab05

08-Sep-16 14:38    <DIR>        .
08-Sep-16 14:38    <DIR>        ..
08-Sep-16 14:38           4,611 access_host_1.txt
07-Sep-16 02:46             407 code_from_postman.py
08-Sep-16 14:30           2,643 our_app.py
08-Sep-16 14:26           2,643 our_code.py
                4 File(s)     10,304 bytes
                2 Dir(s)  112,129,613,824 bytes free

F:\Users\armartirosyan\DevNet Express\lab05>
```

- ファイルの内容を確認するには、任意のテキスト エディタを使用してテキスト ファイルを開くか、ファイルの内容をターミナル ウィンドウに出力します。ここではテキスト エディタを使用してファイルの内容を確認します。



- アプリケーションが正しく機能していることがわかります。

アプリケーションの改善

出力ファイルには、access_host_1.txt という名前を付けるよりも、設定を取得したデバイスのホスト名を付けるほうがわかりやすくなります。さらにファイル名に日付と時刻を加えることで、設定がいつ保存されたが明らかになります。この方法で、ネットワーク全体の設定が含まれたバックアップ ファイルを数秒で作成し、それを変更管理メカニズムとして使用して、特定の時点で特定のデバイスにどのような変更があったかを確認できます。

この方法を実施するには、コードを少し変更する必要があります。

1. `datetime` と `re` の 2 つの新しいモジュールをインポートします。**re** モジュールはデバイスのホスト名の取得に役立ち、**datetime** モジュールでは現在の日付と時刻が得られ、ファイル名にタイムスタンプを付加することができます。
2. `#Import necessary modules`
3. `import requests`
4. `import json`
5. `import datetime`
6. `import re`
7. `get_config()` 関数を変更して、デバイスのホスト名を取得し、ファイル名を目的の形式に変更します。デバイスのホスト名を取得する方法は複数ありますが、正規表現を作成する方法が最も効率的です。変更された関数は次のようになります。
8. `def get_config(token, url, id):`
- 9.
10. `#Define API Call.To get specific device's configuration`
11. `#we will need to add device's ID in the API call`
12. `api_call = "/network-device/" + id + "/config"`
- 13.
14. `#Header information`
15. `headers = {"X-AUTH-TOKEN" : token}`
- 16.
17. `# Combine URL, API call variables`
18. `url +=api_call`
- 19.
20. `response = requests.get(url, headers=headers, verify=False).json()`
- 21.
22. `#Find the hostname in the response body and save it to a hostname variable`
23. `hostname = re.findall('hostname\s(.+)\s', response['response'])[0]`
- 24.
25. `#Create a date_time variable which will hold current time`
26. `date_time = datetime.datetime.now()`
- 27.
28. `#Create a variable which will hold the hostname combined with the date and time`
29. `#The format will be hostname_year_month_day_hour.minute.second`
30. `file_name = hostname + '_' + str(date_time.year) + '_' + str(date_time.month) + '_' + str(date_time.day) + '_' + str(date_time.hour) + '.' + str(date_time.minute) + '.' + str(date_time.second)`
- 31.
32. `file = open(file_name+'.txt', 'w')`

```
33.  
34.  #Write response body to the file  
35.  file.write(response['response'])  
36.  
37.  #Close the file when writing is complete  
38.  file.close()
```

39. ここで Python スクリプトに必要な変更を加えて、コードを再度実行します。

おめでとうございます。デバイスの設定をバックアップしてファイルに保存するアプリケーションを記述しました。