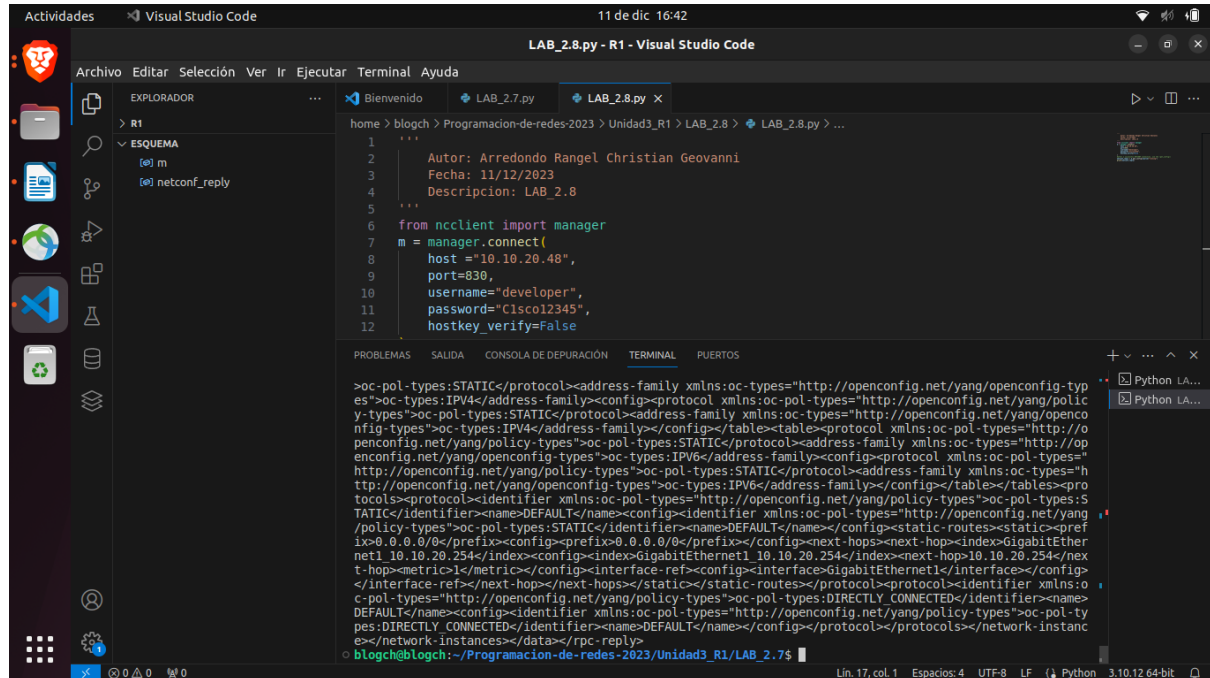
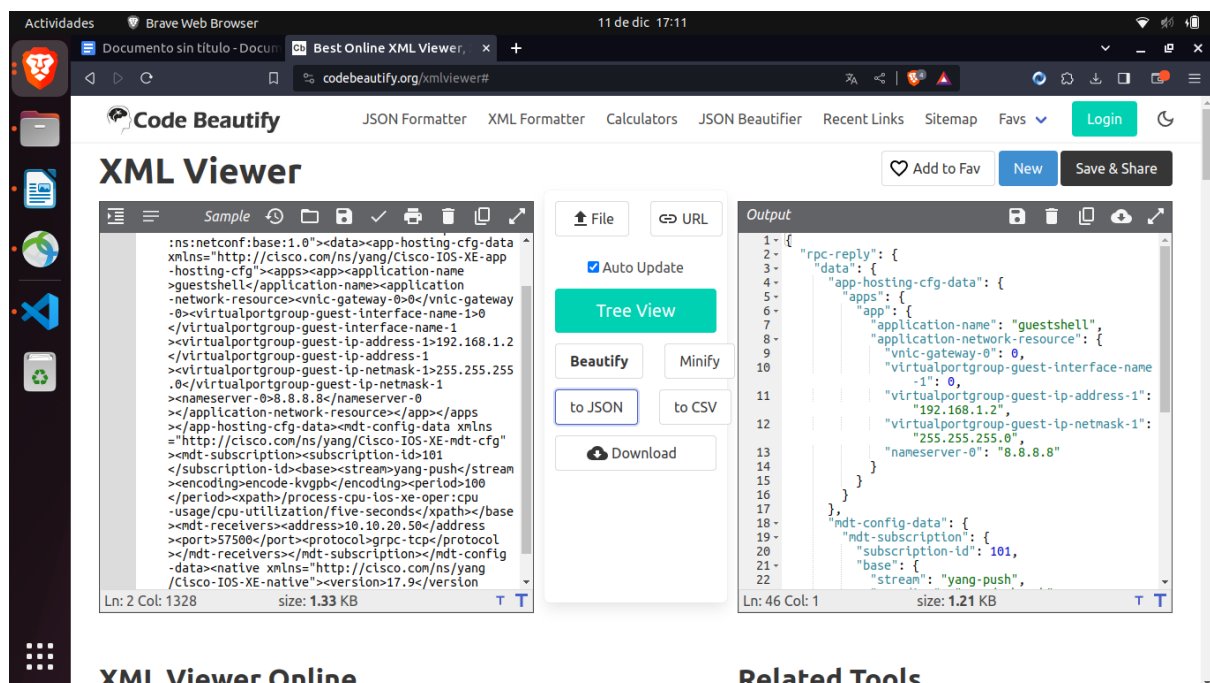


Christian Geovanni Arredondo Rangel  
1222100691  
LAB\_2.8

## Part 1: Retrieve the IOS XE VM's Existing Running Configuration



```
1  ...
2  Autor: Arredondo Rangel Christian Geovanni
3  Fecha: 11/12/2023
4  Descripcion: LAB_2.8
5  ...
6  from ncclient import manager
7  m = manager.connect(
8      host="10.10.20.48",
9      port=830,
10     username="developer",
11     password="Cisco12345",
12     hostkey_verify=False
```



**XML Viewer**

Ln: 2 Col: 1328 size: 1.33 KB

**Output**

```
1- {
2-   "rpc-reply": {
3-     "data": {
4-       "app-hosting-cfg-data": {
5-         "apps": {
6-           "app": {
7-             "application-name": "guestshell",
8-             "application-network-resource": {
9-               "virtualportgroup-guest-interface-name":
10-                "1": 0,
11-               "virtualportgroup-guest-ip-address-1":
12-                "192.168.1.2",
13-               "virtualportgroup-guest-ip-netmask-1":
14-                "255.255.255.0",
15-               "nameserver-0": "8.8.8.8"
16-             }
17-           }
18-         },
19-         "mdt-config-data": {
20-           "mdt-subscription": {
21-             "subscription-id": 101,
22-             "base": {
23-               "stream": "yang-push",
```

**XML Viewer Online**

**Related Tools**

Use CodeBeautify.com to evaluate the response.

Visual Studio Code interface showing the initial setup of a Python script for NetCONF. The Explorer sidebar shows a file named 'netconf\_reply'. The main editor displays a Python script 'LAB\_2.8.py' with code for connecting to a device and retrieving configuration. The Terminal shows the output of the script, displaying XML data for static routes and protocols.

```
LAB_2.8.py - R1 - Visual Studio Code
```

```
10 port=830,
11 username="developer",
12 password="Cisco12345",
13 hostkey_verify=False
14 )
15 #after a successful NETCONF connection, use the "get_config()"
16 netconf_reply = m.get_config(source="running")
17 print(netconf_reply)
18
19
20 print( xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml() )
```

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
```

```
</static>
</static-routes>
</protocol>
</protocol>
<identifier xmlns:oc-pol-types="http://openconfig.net/ya
ng/policy-types">oc-pol-types:DIRECTLY_CONNECTED</identifier>
<name>DEFAULT</name>
<config>
<identifier xmlns:oc-pol-types="http://openconfi
g.net/yang/policy-types">oc-pol-types:DIRECTLY_CONNECTED</identifier>
<name>DEFAULT</name>
</config>
</protocol>
</protocols>
</network-instance>
</network-instances>
</data>
</rpc-reply>
```

Lin. 19, col. 1 Espacios: 4 UTF-8 LF Python 3.10.12 64-bit

Visual Studio Code interface showing the updated Python script for NetCONF. The Explorer sidebar shows two files: 'netconf\_reply' and 'netconf\_filter'. The main editor displays a Python script 'LAB\_2.8.py' with code for connecting to a device and retrieving configuration with a filter. The Terminal shows the output of the script, displaying XML data for static routes and protocols.

```
LAB_2.8.py - R1 - Visual Studio Code
```

```
17 # Obtén la configuración en ejecución sin filtro
18 netconf_reply = m.get()
19 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
20
21 # Intenta obtener la configuración específica usando el filtro en la operación get
22 netconf_filter = """
23 <filter>
24   <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" />
25 </filter>
26 """
27 netconf_reply = m.get(filter=netconf_filter)
28 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
29
```

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
```

```
Python-LAB_2.7 + - ^ X
```

```
">oc-pol-types:DIRECTLY_CONNECTED</identifier> <identifier xmlns:oc-pol-types="http://openconfig.net/yang/policy-types
<name>DEFAULT</name>
<config>
<identifier xmlns:oc-pol-types="http://openconfig.net/yang/poli
cy-types">oc-pol-types:DIRECTLY_CONNECTED</identifie
<name>DEFAULT</name>
</config>
</protocol>
</protocols>
```

Lin. 29, col. 1 Espacios: 4 UTF-8 LF Python 3.10.12 64-bit

## Part 2: Update the Device's Configuration

Visual Studio Code interface showing a Python script (LAB\_2.8\_Parte2.py) and its execution output in the terminal. The script is a RESTCONF client using the ncclient library to interact with a Cisco IOS-XE device. The terminal output shows a successful GET request to the configuration of the 'GigabitEthernet1' interface, returning XML data. The script is located at /home/blogch/Programacion-de-redes-2023/Unidad3\_R1/LAB\_2.8/LAB\_2.8\_Parte2.py.

```
python3 LAB_2.8_Parte2.py
File "/home/blogch/Programacion-de-redes-2023/Unidad3_R1/LAB_2.8/LAB_2.8_Parte2.py", line 28, in <module>
  netconf_reply = m.edit_config(target='running', config=netconf_data)
File "/home/blogch/.local/lib/python3.10/site-packages/ncclient/manager.py", line 257, in execute
  return cls(self._session,
File "/home/blogch/.local/lib/python3.10/site-packages/ncclient/operations/edit.py", line 76, in request
  return self._request(node)
File "/home/blogch/.local/lib/python3.10/site-packages/ncclient/operations/rpc.py", line 375, in _request
  raise self._reply.error
ncclient.operations.rpc.RPCError: {'type': 'protocol', 'tag': 'unknown-element', 'app_tag': None, 'severity': 'error',
'info': '<?xml version='1.0' encoding='UTF-8'?><error-info xmlns='urn:ietf:params:xml:ns:netconf:base:1.0' xmlns:nc='urn:ietf:params:xml:ns:netconf:base:1.0'><bad-element-config/></error-info></n' /rpc/edit-co
nfig\n', 'message': None}
blogch@blogch:~/Programacion-de-redes-2023/Unidad3_R1/LAB_2.8$
```

## EL link ya no funciona

