

NOMBRE: Isidro Lara Lopez

CARRERA: INFRAESTRUCTURA DE REDES DIGITALES

DOCENTE: BARRON RODRIGUEZ GABRIEL

NO. CONTROL: 1221100381

FECHA: 14 DE DICIEMBRE DEL 2022

GRUPO: GIR0441

Lab – NETCONF w/Python: Device Configuration

Objectives

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

Part 2: Update the Device's Configuration

Background / Scenario

In this lab, you will learn how to use the NETCONF ncclient to retrieve the device's configuration, and update and create a new interface configuration. You will also learn why the transactional support of NETCONF is important for getting consistent network changes.

Required Resources

- Access to a router with the IOS XE operating system version 16.6 or higher
- Python 3.x environment

Instructions

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

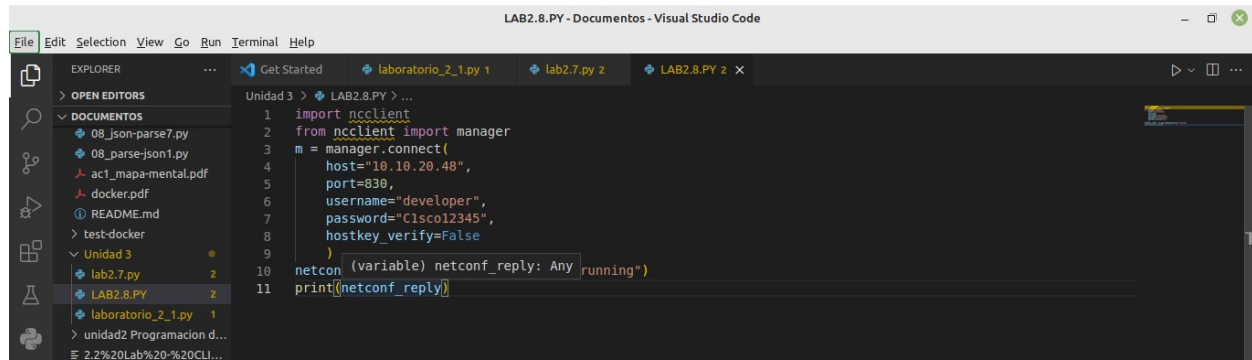
In this part, you will use the ncclient module to retrieve the device's running configuration. The data are returned back in XML form. In the following steps, this data will be transformed into a more human readable format.

Step 1: Use ncclient to retrieve the device's running configuration.

The ncclient module provides a “manager” class with “connect ()” function to setup the remote NETCONF connection. After a successful connection, the returned object represents the NETCONF connection to the remote device.

- In Python IDLE, create a new Python script file:
- In the new Python script file editor, import the “manager” class from the ncclient module:

```
from ncclient import manager
```
- Using the manager .connect () function, set up an m connection object to the IOS XE device.

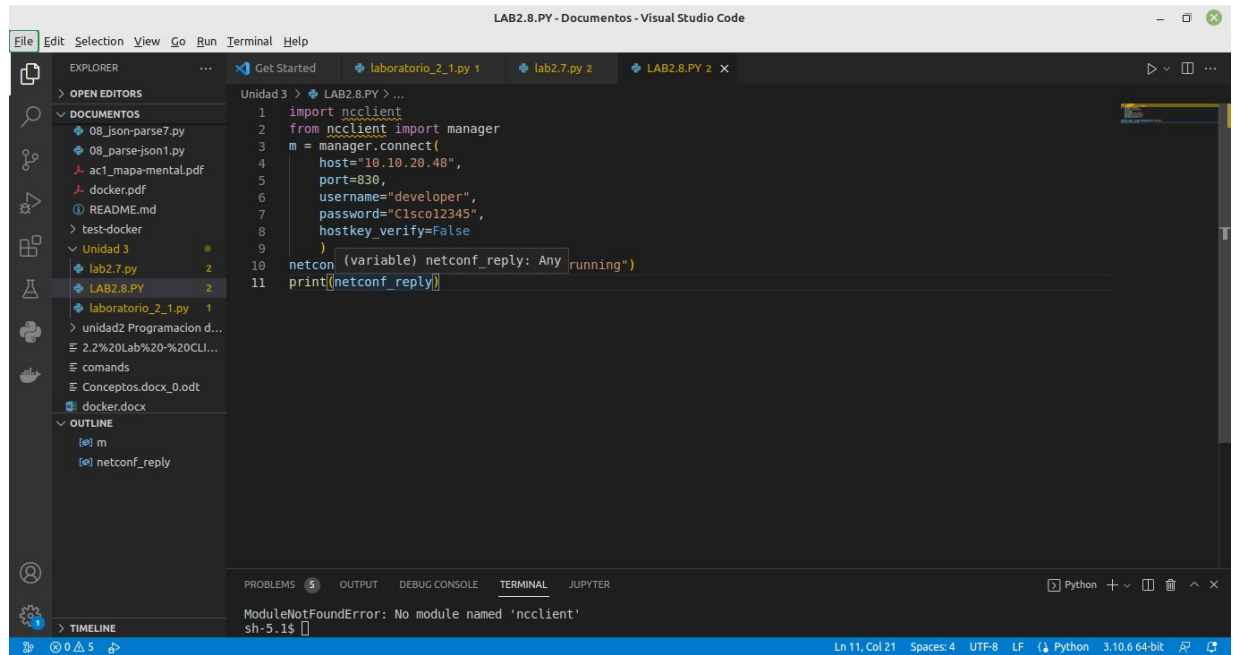


The parameters of the `manager.connect()` function are:

- **host** – This is the address (host or IP) of the remote device (Adjust the IP address to match the router's current address.).
 - **port** – This is the remote port of the SSH service.
 - **username** – This is the remote SSH username (In this lab, use "cisco" because that was set up in the IOS XE VM.).
 - **password** – This is the remote SSH password (In this lab, use "cisco123!" because that was set up in the IOS XE VM.).
 - **hostkey_verify** – Use this to verify the SSH fingerprint (In this lab, it is safe to set to False; however, in production environments you should always verify the SSH fingerprints.).
- d. After a successful NETCONF connection, use the `get_config()` function of the **"m"** NETCONF session object to retrieve and print the device's running configuration. The `get_config()` function expects a "source" string parameter that defines the source NETCONF data-store.

```
netconf_reply = m.get_config(source="running")
print(netconf_reply)
```

Lab – NETCONF w/Python: Device Configuration



- e. Execute the Python script and explore the output.

Step 2: Use CodeBeautify.com to evaluate the response.

Code Beautify maintains a website for viewing code in a more human readable format. The XML viewer URL is <https://codebeautify.org/xmlviewer>

- Copy the XML from IDLE to XML Viewer.
- Click **Tree View** or **Beautify / Format** to render the raw XML output into a more human readable format.
- To simplify the view, close the XML elements that are under the rpc-reply/data structure.
- Note that the opened rpc-reply/data/native element contains an attribute xmlns that points to “Cisco-IOS-XE-native” YANG model. That means this part of the configuration is Cisco Native for IOS XE.
- Also note that there are two “interfaces” elements. The one with xmlns is pointing to the “http://openconfig.net/yang/interfaces” YANG model, while the other is pointing to the “ietf-interfaces” YANG model.

Both are used to describe the configuration of the interfaces. The difference is that the openconfig.net YANG model does support sub-interfaces, while the ietf-interfaces YANG model does not.

The screenshot shows the CodeBeautify website interface. On the left, the 'XML Input' section contains a large block of raw XML code. In the center, there are buttons for 'Load Url', 'Browse', 'Tree View', and a prominent green 'OPEN' button. On the right, the 'Result : Beautify XML' section displays the same XML code formatted into a clean, human-readable structure with proper indentation and line numbers. The beautified XML shows a root element 'rpc-reply' containing a 'data' element with a 'native' configuration block. The 'native' block includes various configuration elements like 'loopback', 'interface', and 'router'.

Utilizamos codebeautify para pasar la respuesta xml a un texto mas legible para nosotros los humanos y este fue el resultado.

Step 3: Use `toprettyxml()` function to prettify the output.

- a. Python has built in support to work with XML files. The “`xml.dom.minidom`” module can be used to prettify the output with the `toprettyxml()` function.
- b. Import the “`xml.dom.minidom`” module:

```
import xml.dom.minidom
```
- c. Replace the simple print function “`print(netconf_reply)`” with a version that prints prettified XML output:

```
print( xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml() )
```
- d. Execute the updated Python script and explore the output.


```
ilara@LPI: ~/Documentos/Unidad 3
Archivo Editar Ver Buscar Terminal Ayuda
ilara@LPI:~/Documentos/Unidad 3 $ python3 LAB2.8.PY
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:aaf28855-8b23-48ee-8fc8-86f8fdb02c2b">
  <data>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <version>16.11</version>
      <boot-start-marker/>
      <boot-end-marker/>
      <banner>
        <motd>
          <banner>^C</banner>
        </motd>
      </banner>
      <memory>
        <free>
          <low-watermark>
            <processor>80557</processor>
          </low-watermark>
        </free>
      </memory>
      <call-home>
        <contact-email-addr xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">sch-smart-licensing@cisco.com</contact-email-addr>
        <profile xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
          <profile-name>CiscoTAC-1</profile-name>
          <active>true</active>
        </profile>
      </call-home>
      <service>
        <timestamps>
          <debug>
            <datetime>
              <msec/>
            </datetime>
          </debug>
          <log>
            <datetime>
              <msec/>
            </datetime>
          </log>
        </timestamps>
      </service>
    </native>
  </data>
</rpc-reply>
```

```
ilara@LPI: ~/Documentos/Unidad 3
Archivo Editar Ver Buscar Terminal Ayuda
</ip>
<interface>
  <GigabitEthernet>
    <name>1</name>
    <description>MANAGEMENT INTERFACE - DON'T TOUCH ME</description>
    <ip>
      <address>
        <primary>
          <address>10.10.20.48</address>
          <mask>255.255.255.0</mask>
        </primary>
      </address>
    </ip>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>2</name>
    <description>Network Interface</description>
    <shutdown/>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>3</name>
    <description>Network Interface</description>
    <shutdown/>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
  </GigabitEthernet>
</interface>
```



```
ilara@LPI: ~/Documentos/Unidad 3
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

<exec-default>deny</exec-default>
<enable-external-groups>true</enable-external-groups>
<rule-list>
  <name>admin</name>
  <group>PRIV15</group>
  <rule>
    <name>permit-all</name>
    <module-name>*</module-name>
    <access-operations>*</access-operations>
    <action>permit</action>
  </rule>
</rule-list>
</nacm>
<routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <routing-instance>
    <name>default</name>
    <description>default-vrf [read-only]</description>
    <routing-protocols>
      <routing-protocol>
        <type>static</type>
        <name>1</name>
        <static-routes>
          <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing">
            <route>
              <destination-prefix>0.0.0.0</destination-prefix>
              <next-hop>
                <outgoing-interface>GigabitEthernet1</outgoing-interface>
              </next-hop>
            </route>
          </ipv4>
        </static-routes>
      </routing-protocol>
    </routing-protocols>
  </routing-instance>
</routing>
</data>
</rpc-reply>
ilara@LPI:~/Documentos/Unidad 3 $
```

Esta es la salida que muestro al ejecutar la nueva salida con un formato mas legible para los humanos con el `print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())`

Step 4: Use filters to retrieve a configuration defined by a specific YANG model.

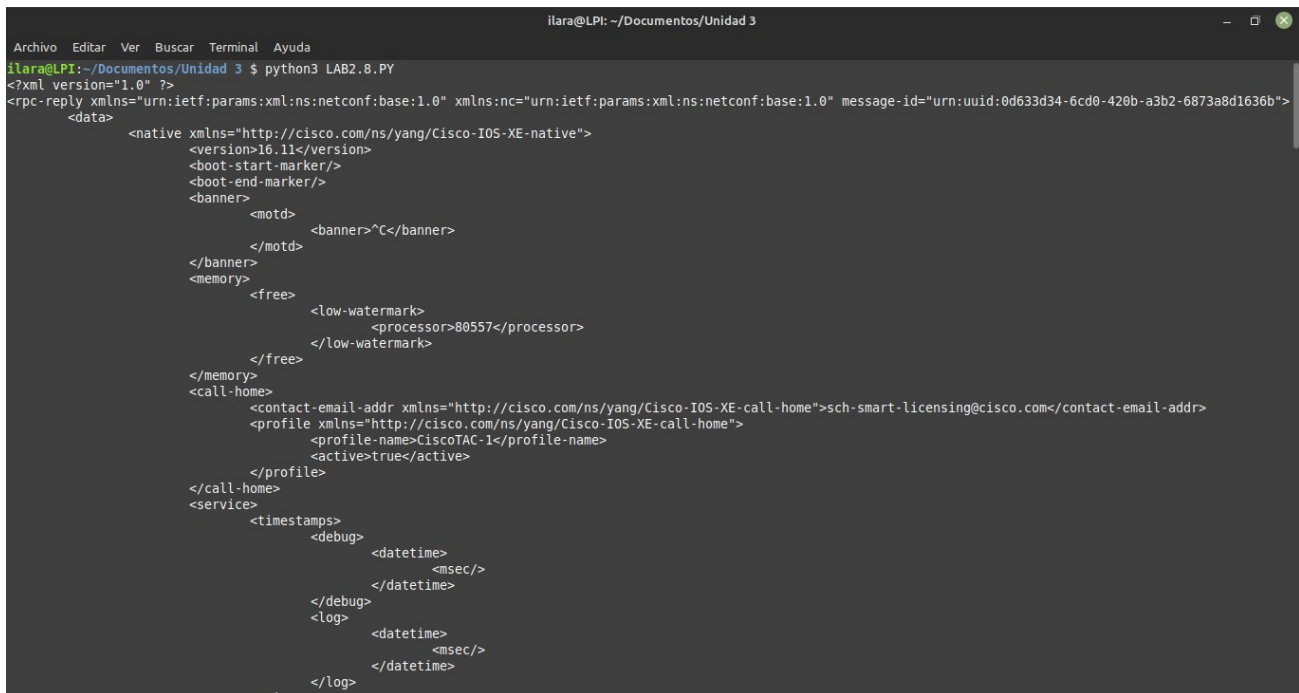
- NETCONF has support to return only data that are defined in a filter element.
- Create the following `netconf_filter` variable containing an XML NETCONF filter element that is designed to retrieve only data that is defined by the Cisco IOS XE Native YANG model:

```
netconf_filter = """
<filter>
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" />
</filter>
"""
```

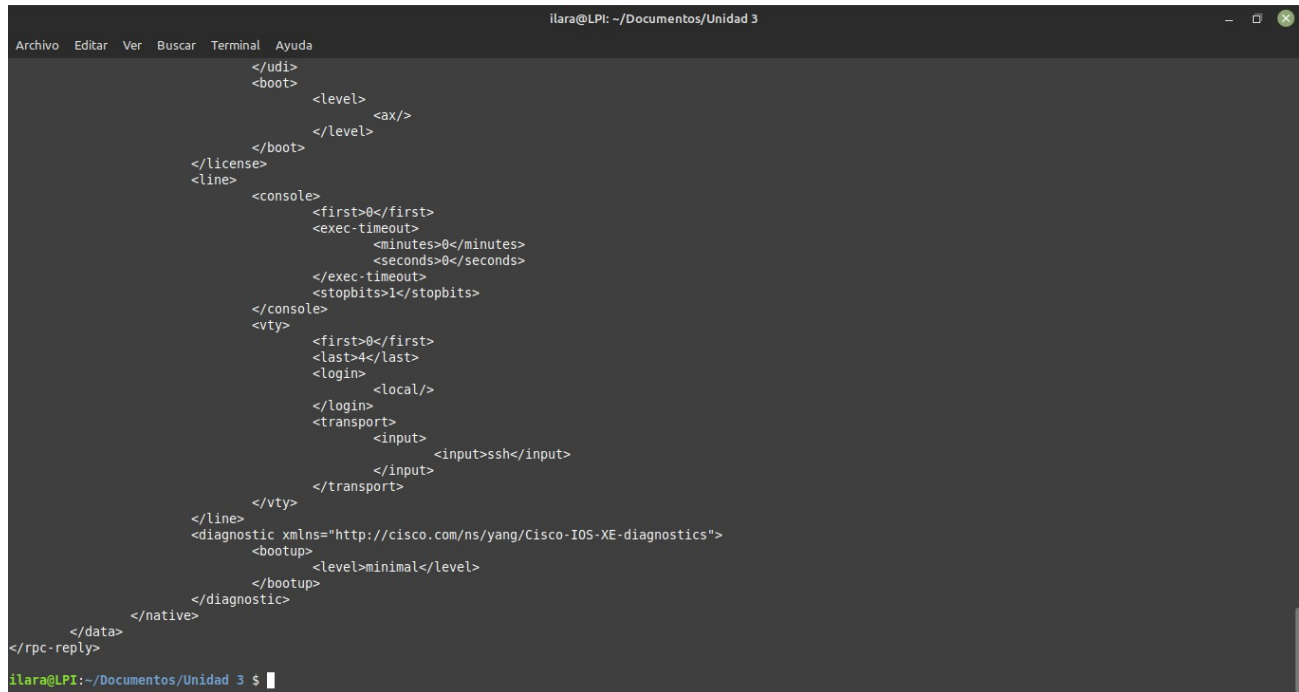
- Include the `netconf_filter` variable in the `get_config()` call using the “filter” parameter:

```
netconf_reply = m.get_config(source="running", filter=netconf_filter)
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

- Execute the updated Python script and explore the output



```
ilara@LPI: ~/Documentos/Unidad 3
ilara@LPI:~/Documentos/Unidad 3 $ python3 LAB2.8.PY
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:0d633d34-6cd0-420b-a3b2-6873a8d1636b">
  <data>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <version>16.11</version>
      <boot-start-marker/>
      <boot-end-marker/>
      <banner>
        <motd>
          <banner>^C</banner>
        </motd>
      </banner>
      <memory>
        <free>
          <low-watermark>
            <processor>80557</processor>
          </low-watermark>
        </free>
      </memory>
      <call-home>
        <contact-email-addr xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">sch-smart-licensing@cisco.com</contact-email-addr>
        <profile xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
          <profile-name>CiscoTAC-1</profile-name>
          <active>true</active>
        </profile>
      </call-home>
      <service>
        <timestamps>
          <debug>
            <datetime>
              <msec/>
            </datetime>
          </debug>
          <log>
            <datetime>
              <msec/>
            </datetime>
          </log>
        </timestamps>
      </service>
    </native>
  </data>
</rpc-reply>
```



```
ilara@LPI: ~/Documentos/Unidad 3
Archivo  Editar  Ver  Buscar  Terminal  Ayuda

</udi>
<boot>
  <level>
    <ax/>
  </level>
</boot>
</license>
<line>
  <console>
    <first>0</first>
    <exec-timeout>
      <minutes>0</minutes>
      <seconds>0</seconds>
    </exec-timeout>
    <stopbits>1</stopbits>
  </console>
  <vty>
    <first>0</first>
    <last>4</last>
    <login>
      <local/>
    </login>
    <transport>
      <input>
        <input>ssh</input>
      </input>
    </transport>
  </vty>
</line>
<diagnostic xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-diagnostics">
  <bootup>
    <level>minimal</level>
  </bootup>
</diagnostic>
</native>
</data>
</rpc-reply>
ilara@LPI:~/Documentos/Unidad 3 $
```

esta es la nueva salida que nos s al craar el nuevo filtro para recuperar solo datos definidos por el modelo YANG nativo de Cisco IOS XE

Part 2: Update the Device's Configuration

Step 1: Create a new Python script file.

- In IDLE, create a new Python script file.
- Import the required modules and set up the NETCONF session:

```
from ncclient import manager
import xml.dom.minidom

m = manager.connect(
    host="192.168.56.101",
    port=830,
    username="cisco",
    password="cisco123!",
    hostkey_verify=False
)
```

Step 2: Change the hostname.

- f. In order to update an existing setting in the configuration, you can extract the setting location from the configuration retrieved in Step 1.
- g. The configuration update is always enclosed in a “config” XML element. This element includes a tree of XML elements that require updates.
- h. Create a **netconf_data** variable that holds a configuration update for the hostname element as defined in the Cisco IOS XE Native YANG Model:

```
netconf_data = """
<config>
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <hostname>NEWHOSTNAME</hostname>
  </native>
</config>
"""
```

- i. Edit the existing device configuration with the “edit_config()” function of the “m” NETCONF session object. The edit_config() function expects two parameters:
 - **target** – This is the target netconf data-store to be updated.
 - **config** – This is the configuration update.

The edit_config() function returns an XML object containing information about the success of the change. After editing the configuration, print the returned value:

```
netconf_reply = m.edit_config(target="running", config=netconf_data)
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

- j. Before executing the new Python script, check the current hostname by connecting to the console of the IOS XE VM.
- k. Execute the Python script and explore the output.
- l. After executing the Python script, if the reply contained the <ok/> element, verify whether the current hostname has been changed by connecting to the console of the IOS XE VM.

```
ilara@LPI: ~/Documentos/Unidad 3
Archivo Editar Ver Buscar Terminal Ayuda
ilara@LPI:~/Documentos/Unidad 3 $ cd python3 lab^C
ilara@LPI:~/Documentos/Unidad 3 $ ls -l
total 16
-rw-r--r-- 1 ilara ilara 298 dic 13 23:01 lab2.7.py
-rw-r--r-- 1 ilara ilara 518 dic 14 02:22 lab2.8p4.py
-rw-r--r-- 1 ilara ilara 606 dic 14 02:14 LAB2.8.PY
-rw-r--r-- 1 ilara ilara 454 dic 13 20:23 laboratorio_2_1.py
ilara@LPI:~/Documentos/Unidad 3 $ python3 lab2.8p4.py
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:92797b64-adc7-43f8-a31c-5aecc47190fa">
  <ok/>
</rpc-reply>
ilara@LPI:~/Documentos/Unidad 3 $
```

La respuesta fu ok que es la respuesta esperada a recibir esto quiere deccir que se canbio el nombre del host

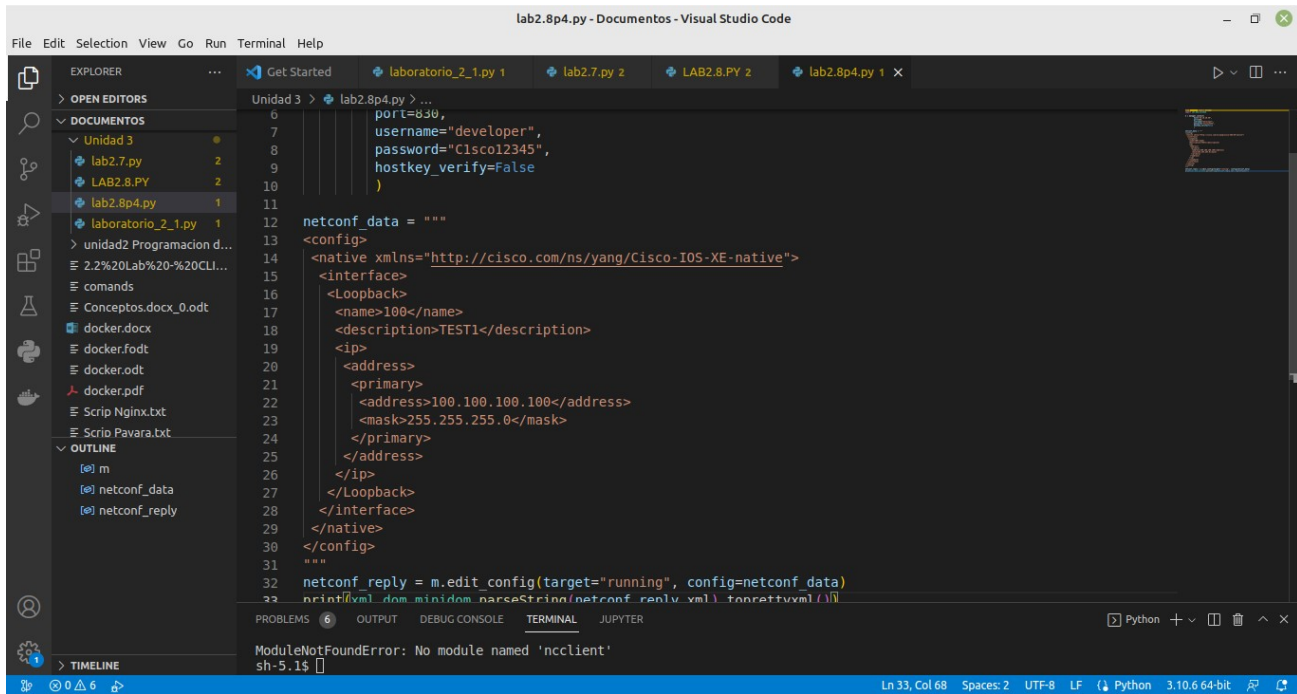
Step 3: Create a loopback interface

- m. Update the **netconf_data** variable to hold a configuration update that creates a new loopback **100** interface with the IP address 100.100.100.100/24:

```
netconf_data = """
<config>
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <interface>
      <Loopback>
        <name>100</name>
        <description>TEST1</description>
        <ip>
          <address>
            <primary>
              <address>100.100.100.100</address>
              <mask>255.255.255.0</mask>
            </primary>
          </address>
        </ip>
      </Loopback>
    </interface>
  </native>
</config>
"""
```

- n. Add the new loopback 100 interface by editing the existing device configuration using the "edit_config()" function:
- ```
netconf_reply = m.edit_config(target="running", config=netconf_data)
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```
- o. Before executing the updated Python script, check the existing loopback interface by connecting to the console of the IOS XE VM using the **show ip int brief** and **show int desc** commands.
- p. Execute the Python script and explore the output
- q. After executing the Python script, if the reply contained the <ok/> element, verify whether the current loopback interfaces have changed by connecting to the console of the IOS XE VM.

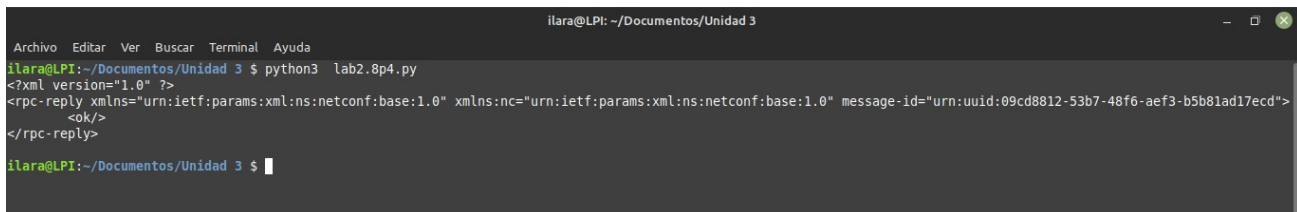
## Lab – NETCONF w/Python: Device Configuration



The screenshot shows the Visual Studio Code editor with a file named `lab2.8p4.py` open. The script defines a `netconf_data` variable containing a NETCONF configuration for a loopback interface named `100`. The configuration includes a description `TEST1` and an IP address `100.100.100.100/32`. The script also shows the `netconf_reply` variable being assigned the result of a `netconf.edit_config` call. The terminal at the bottom shows an error: `ModuleNotFoundError: No module named 'ncclient'`.

```
lab2.8p4.py - Documentos - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
 OPEN EDITORS
 DOCUMENTOS
 Unidad 3
 lab2.7.py
 LAB2.8.PY
 lab2.8p4.py
 laboratorio_2_1.py
 unidad2 Programacion d...
 2.2%20Lab%20-%20CLI...
 comands
 Conceptos.docx_0.odt
 docker.docx
 docker.fodt
 docker.odt
 docker.pdf
 Scrip Nginx.txt
 Scrio Pavana.txt
 OUTLINE
 m
 netconf_data
 netconf_reply
 TIMELINE
 Get Started
 laboratorio_2_1.py
 lab2.7.py
 LAB2.8.PY
 lab2.8p4.py
 Unidad 3 > lab2.8p4.py > ...
 6 port=830,
 7
 8 username="developer",
 9 password="Cisco12345",
 10 hostkey_verify=False
 11)
 12
 13 netconf_data = ""
 14 <config>
 15 <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
 16 <interface>
 17 <Loopback>
 18 <name>100</name>
 19 <description>TEST1</description>
 20 <ip>
 21 <address>
 22 <primary>
 23 <address>100.100.100.100</address>
 24 <mask>255.255.255.0</mask>
 25 </primary>
 26 </address>
 27 </ip>
 28 </Loopback>
 29 </interface>
 30 </native>
 31 </config>
 32
 33 netconf_reply = m.edit_config(target="running", config=netconf_data)
 34 print(xml.dom.minidom.parseString(netconf_reply.xml).tostring())
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
ModuleNotFoundError: No module named 'ncclient'
sh-5.1$
```

Esta es la imagen del escript realizado para crear una interfaz loopback en el host connectado.



The screenshot shows a terminal window with the command `python3 lab2.8p4.py` executed. The output shows an XML response from the device, indicating that the configuration was successfully applied. The response includes a `<ok>` tag within a `<rpc-reply>` tag.

```
ilara@LPI: ~/Documentos/Unidad 3
Archivo Editar Ver Buscar Terminal Ayuda
ilara@LPI:~/Documentos/Unidad 3 $ python3 lab2.8p4.py
<?xml version='1.0' ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:09cd8812-53b7-48f6-aef3-b5b81ad17ecd">
 <ok/>
</rpc-reply>
ilara@LPI:~/Documentos/Unidad 3 $
```

La respuesta que nas marco fue la esperada que era un ok lo que significa que creamos la interfaz correctamente.

### Step 4: Attempt to create a new loopback interface with a conflicting IP address.

- Update the `netconf_data` variable to hold a configuration update that creates a new loopback **111** interface with the same IP address as on loopback 100: 100.100.100.100/32:

```
netconf_data = ""
<config>
 <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
 <interface>
 <Loopback>
 <name>111</name>
```



```
<description>TEST1</description>
<ip>
 <address>
 <primary>
 <address>100.100.100.100</address>
 <mask>255.255.255.0</mask>
 </primary>
 </address>
</ip>
</Loopback>
</interface>
</native>
</config>
"""
```

- b. Attempt to add the new loopback 111 interface by editing the existing device configuration using the "edit\_config()" function:

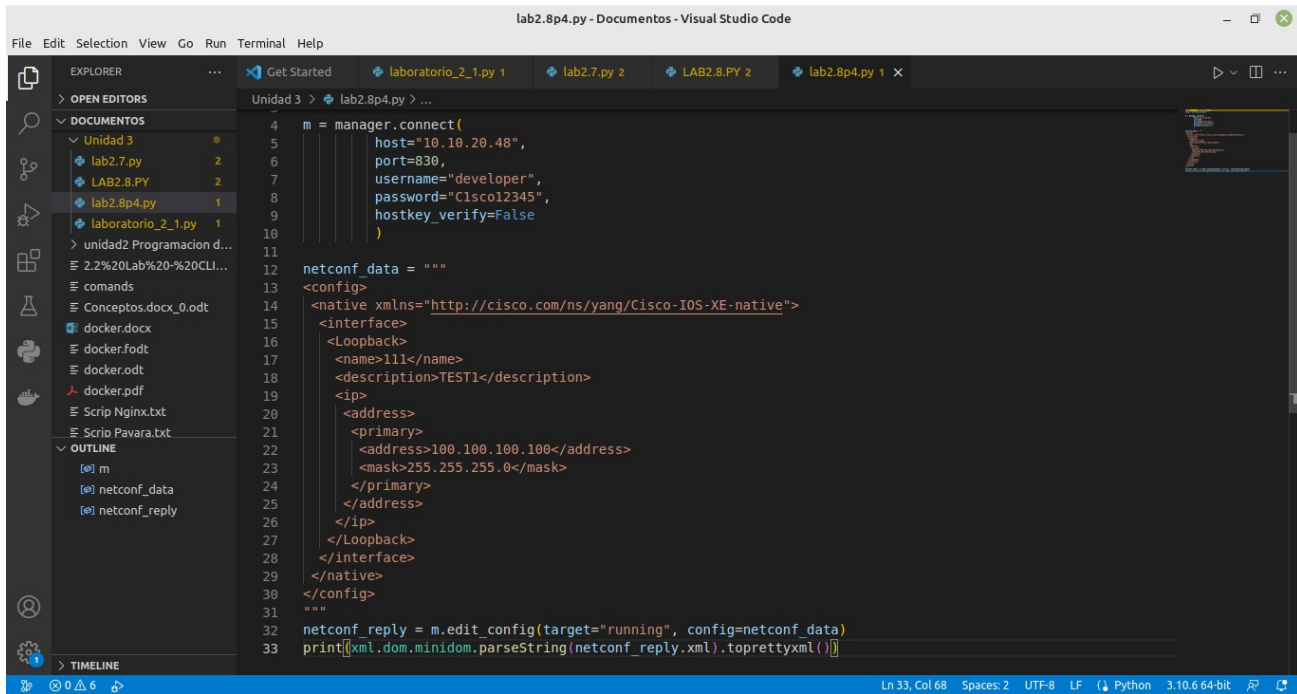
```
netconf_reply = m.edit_config(target="running", config=netconf_data)
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

- c. Before executing the updated Python script, check the existing loopback interface by connecting to the console of the IOS XE VM using the **show ip int brief** and **show int desc** commands.
- d. Execute the Python script and explore the output.
- The device has refused one or more configuration settings. With NETCONF, thanks to the transactional behavior, no partial configuration change has been applied but the whole transaction was canceled.
- e. After executing the Python script, verify that no configuration changes, not even partial, have been applied.

Con este script intentaremos crear una interfaz lpbak que tenga un conflicto.

Y como podemos ver al ejecutar el script la salida marca un error en las líneas del código por que la interfaz no está correctamente creada por tener un error.

## Lab – NETCONF w/Python: Device Configuration



The screenshot shows the Visual Studio Code interface with a Python file named `lab2.8p4.py` open. The file contains a script that connects to a device via NETCONF and sends a configuration. The Explorer sidebar on the left shows a project structure with files like `lab2.7.py`, `LAB2.8.PY`, and `lab2.8p4.py`. The Outline sidebar shows variables `m`, `netconf_data`, and `netconf_reply`. The main editor area shows the following code:

```
4 m = manager.connect(
5 host="10.10.20.48",
6 port=830,
7 username="developer",
8 password="Cisco12345",
9 hostkey_verify=False
10)
11
12 netconf_data = """
13 <config>
14 <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
15 <interface>
16 <Loopback>
17 <name>111</name>
18 <description>TEST1</description>
19 <ip>
20 <address>
21 <primary>
22 <address>100.100.100.100</address>
23 <mask>255.255.255.0</mask>
24 </primary>
25 </address>
26 </ip>
27 </Loopback>
28 </interface>
29 </native>
30 </config>
31 """
32 netconf_reply = m.edit_config(target="running", config=netconf_data)
33 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

The status bar at the bottom indicates the file is at line 33, column 68, using UTF-8 encoding, LF line endings, and the Python 3.10.6 64-bit interpreter.

```
ilara@LPI: ~/Documentos/Unidad 3
Archivo Editar Ver Buscar Terminal Ayuda
ilara@LPI:~/Documentos/Unidad 3 $ python3 lab2.8p4.py
<?xml version="1.0" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:09cd8812-53b7-48f6-aef3-b5b81ad17ecd">
 <ok/>
</rpc-reply>

ilara@LPI:~/Documentos/Unidad 3 $ python3 lab2.8p4.py
Traceback (most recent call last):
 File "/home/ilara/Documentos/Unidad 3 /lab2.8p4.py", line 32, in <module>
 netconf_reply = m.edit_config(target="running", config=netconf_data)
 File "/home/ilara/.local/lib/python3.10/site-packages/ncclient/manager.py", line 246, in execute
 return cls(self._session,
 File "/home/ilara/.local/lib/python3.10/site-packages/ncclient/operations/edit.py", line 76, in request
 return self._request(node)
 File "/home/ilara/.local/lib/python3.10/site-packages/ncclient/operations/rpc.py", line 375, in _request
 raise self._reply.error
ncclient.operations.rpc.RPCError: inconsistent value: Device refused one or more commands
ilara@LPI:~/Documentos/Unidad 3 $
```

En este laboratorio venimos repasando los protocolos que hemos realizado investigación en laboratorios pasados, aquí se siguen ocupando el Netconf y los modelos yang que se utilizan en laboratorios pasados.

En este laboratorio se realizó todo lo que hicimos en los laboratorios pasados con la librería ncclient para conectarse remotamente, y los modelos yang para interpretar el resultado de la petición que realizamos, configuraciones, y su documentación para saber lo que es el Netconf y el ncclient y también para que se utiliza, que se hacen para enviar comandos de configuración a los router remotamente de los cuales. También utilizamos la función `toprettyxml()` la cual nos sirve para embellecer y darle formato a la salida que nos dio, se utilizaron comandos para enviar configuraciones e loopback y en la última no se logró porque se le puso una dirección de red errónea por ello nos marcó un error de ejecución.