

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

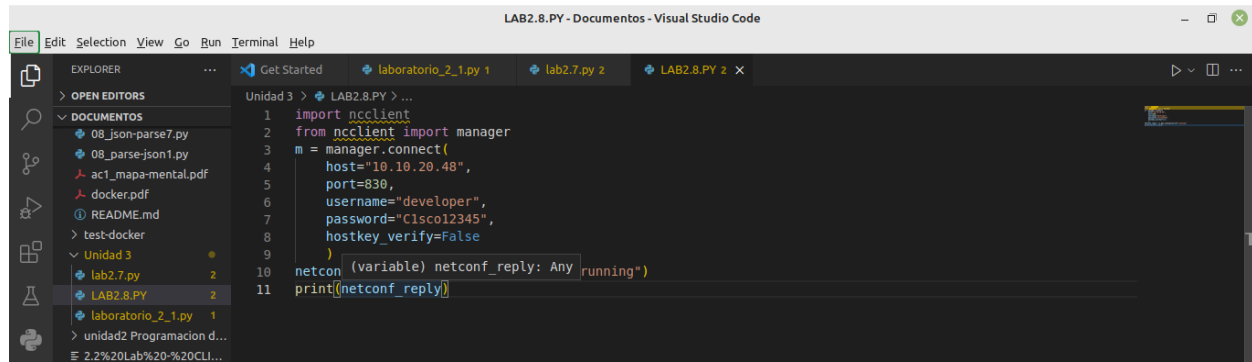
In this part, you will use the ncclient module to retrieve the device's running configuration. The data are returned back in XML form. In the following steps, this data will be transformed into a more human readable format.

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)
```



```

Archivo Editar Ver Buscar Terminal Ayuda
ilara@LPI: ~/Documentos/Unidad 3
faces/><config><enabled=false/><enabled=</ipv6></subinterface/><subinterfaces=</interface/></interfaces><lldp xmlns="http://openconfig.net/yang/lldp"><config>
enabled=false/><enabled=</config></interface><interface-name=GigabitEthernet1/><name=</config><name=GigabitEthernet1/><name=enabled=true/><enabled=</config></interface><int
erface-name=GigabitEthernet2/><name=</config><name=GigabitEthernet2/><name=enabled=true/><enabled=</config></interface><interface-name=GigabitEthernet3/><name=</config><nam
e=GigabitEthernet3/><name=enabled=true/><enabled=</config></interface></interfaces></lldp><network-instances xmlns="http://openconfig.net/yang/network-instance"><network-
instance-name=default/><name=</config><name=default/><name-type xmlns:oc-ni-types="http://openconfig.net/yang/network-instance-types"><oc-ni-types:DEFAULT_INSTANCE/><types>
description=default-vrf [read-only]/<description/></config><tables=table-protocol xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:DIRECTLY_CON
NECTED/></protocol><address-family xmlns:oc-types="http://openconfig.net/yang/openconfig-types"><oc-types:IPv4/><address-family/></config><protocol xmlns:oc-pol-types="http://
openconfig.net/yang/policy-types"><oc-pol-types:DIRECTLY_CONNECTED/></protocol><address-family xmlns:oc-types="http://openconfig.net/yang/openconfig-types"><oc-types:IPv4/><a
dress-family/></config></table=table-protocol xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:DIRECTLY_CONNECTED/></protocol><address-family xm
ls:oc-types="http://openconfig.net/yang/openconfig-types"><oc-types:IPv6/><address-family/></config><protocol xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><o
c-pol-types:DIRECTLY_CONNECTED/></protocol><address-family xmlns:oc-types="http://openconfig.net/yang/openconfig-types"><oc-types:IPv6/><address-family/></config></table=tab
le-protocol xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:STATIC/></protocol><address-family xmlns:oc-types="http://openconfig.net/yang/openco
nfig-types"><oc-types:IPv4/><address-family/></config><protocol xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:STATIC/></protocol><address-fam
ily xmlns:oc-pol-types="http://openconfig.net/yang/openconfig-types"><oc-types:IPv4/><address-family/></config></table=table-protocol xmlns:oc-pol-types="http://openconfig.net/yang
/policy-types"><oc-pol-types:STATIC/></protocol><address-family xmlns:oc-types="http://openconfig.net/yang/openconfig-types"><oc-types:IPv6/><address-family/></config></protocol
xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:STATIC/></protocol><address-family xmlns:oc-types="http://openconfig.net/yang/openconfig-types">
<oc-types:IPv6/><address-family/></config></table=tables></protocols></protocol><identifier xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:STATIC
/><identifier-name=DEFAULT/><name=</config><identifier xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:STATIC/><identifier-name=DEFAULT/><name=</
config><static-routes static-prefix=0.0.0.0/8/</prefix><config><prefix=0.0.0.0/8/</prefix></config><next-hops next-hop=<index><GigabitEthernet1 10.10.20.254/</index></config>
g-next-hop=<next-hop/><static/></static-routes></protocol></protocol><identifier xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:DIRECTLY
CONNECTED/><identifier-name=DEFAULT/><name=</config><identifier xmlns:oc-pol-types="http://openconfig.net/yang/policy-types"><oc-pol-types:DIRECTLY_CONNECTED/><identifier->
name=DEFAULT/><name=</config></protocol></protocols></network-instances></network-instances></interfaces xmlns="urn:ietf:params.xml:yang:iana-if-type"><interface-name=
GigabitEthernet1/><name=</description><MANAGEMENT INTERFACE - DON'T TOUCH ME/><description/><type xmlns:ianaif="urn:ietf:params.xml:yang:iana-if-type"><ianaif:ethernetCsmacd/>
<type=enabled=true/><enabled=ip4 xmlns="urn:ietf:params.xml:yang:ietf-ip"><address=ip=10.10.20.48/><ipv6-netmask=255.255.255.0/></netmask></address></ipv4=ipv6
xmlns="urn:ietf:params.xml:yang:ietf-ip"><ip=ipv6/></interface></interface-name=GigabitEthernet2/><name=</description><Network Interface/><description/><type xmlns:ianaif="ur
n:ietf:params.xml:yang:iana-if-type"><ianaif:ethernetCsmacd/><type=enabled=false/><enabled=ip4 xmlns="urn:ietf:params.xml:yang:ietf-ip"><ip=ipv4/><ipv6 xmlns="urn:i
etf:params.xml:yang:ietf-ip"><ip=ipv6/></interface></interface-name=Loopback1/><name type xmlns:ianaif="urn:ietf:params.xml:yang:iana-if-type"><ianaif:software-loopback/><type=en
abled=true/><enabled=ip4 xmlns="urn:ietf:params.xml:yang:ietf-ip"><address=ip=3.3.3.3/><ipv6-netmask=255.255.255.0/></netmask></address></ipv4=ipv6 xmlns="urn:ietf:par
ams.xml:yang:ietf-ip"><ip=ipv6/></interface></interfaces><nacm xmlns="urn:ietf:params.xml:yang:ietf-netconf-cmd"><enable-nacm><rule=enable-nacm><default-deny/re
ad-default><write-default><deny/write-default><exec-default><deny/exec-default><enable-external-groups><true></enable-external-groups><rule-list><name=admin/><name-group=P
RTV15/><group=rules><name-permit-all/><name-module name=""></module-name><access-operations><access-operations><action-permit/><action/></rule></rule-list></nacm></routing x
mlns="urn:ietf:params.xml:yang:ietf-routing"><routing-info><name=default/><name=</description><default-vrf [read-only]/><description><routing-protocols><routing-proto
cols><type=static/><type=name=1/><name=static-routes><ipv4 xmlns="urn:ietf:params.xml:yang:ietf-ipv4-unicast-routing"><route=destination-prefix=0.0.0.0/8/</destination-
prefix><next-hop outgoing-interface=GigabitEthernet1 outgoing-interface/><next-hop/></route></ipv4></static-routes></routing-protocols></routing-protocols></routing-ins
tances></routing/></data></rpc-reply>
```

En estas imágenes vemos la salida que muestra al ejecutarlo desde la terminal de mi laptop

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>

- f. Copy the XML from IDLE to XML Viewer.
- g. Click **Tree View** or **Beautify / Format** to render the raw XML output into a more human readable format.
- h. To simplify the view, close the XML elements that are under the rpc-reply/data structure.
- i. 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.
- j. 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.

[illegible]

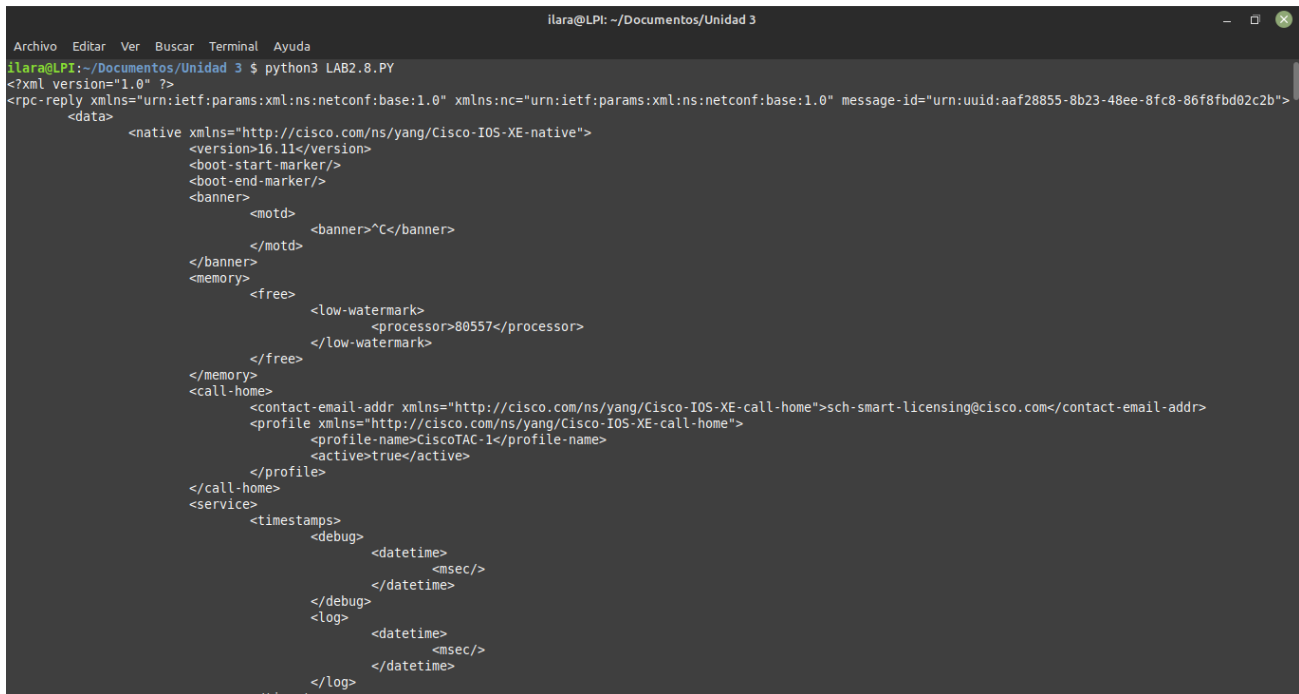
Utilizamos codebeatyfu para pasar la respuesta xml a un texto más legible para nosotros los humanos y este fue el resultado.

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

- 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.
-
- Import the “xml.dom.minidom” module:

```
import xml.dom.minidom
```
- 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() )
```
- 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-8fc8-86f8fbd02c2b">
  <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>
```


Lab – NETCONF w/Python: Device Configuration

```
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>
</data>
</rpc-reply>
```

```
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 mostro al ejecutar la nueva salida con un formato más 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

Lab – NETCONF w/Python: Device Configuration

```
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: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.

- a. In IDLE, create a new Python script file.
- b. 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.

- g. In order to update an existing setting in the configuration, you can extract the setting location from the configuration retrieved in Step 1.
- h. The configuration update is always enclosed in a “config” XML element. This element includes a tree of XML elements that require updates.
- i. 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>
"""
```

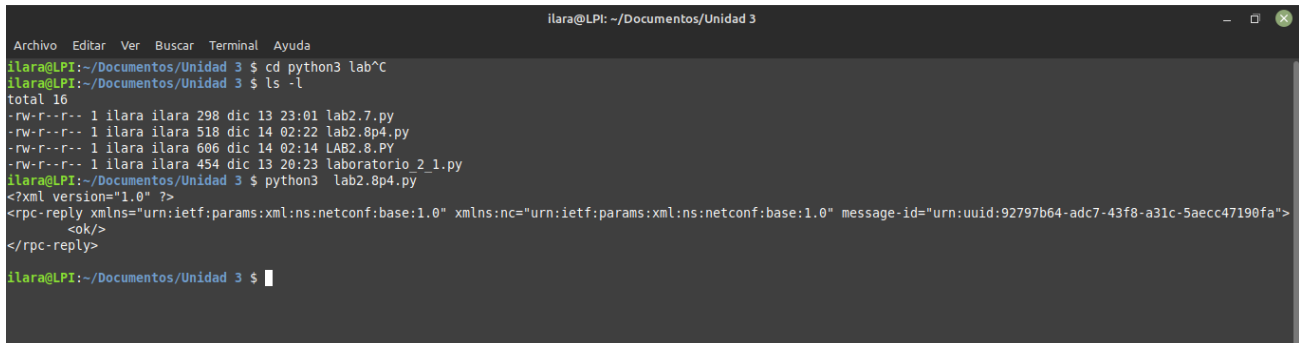
- j. Edit the existing device configuration with the “`edit_config()`” function of the “`m`” NETCONF session object. The `edit_config()` function expects two parameters:
 - **target** – This is the target netconf data-store to be updated.
 - **config** – This is the configuration update.

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())
```

- k. Before executing the new Python script, check the current hostname by connecting to the console of the IOS XE VM.

- I. Execute the Python script and explore the output.
- m. 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-5a6cc47190fa">
  <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

- n. 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>
"""
```

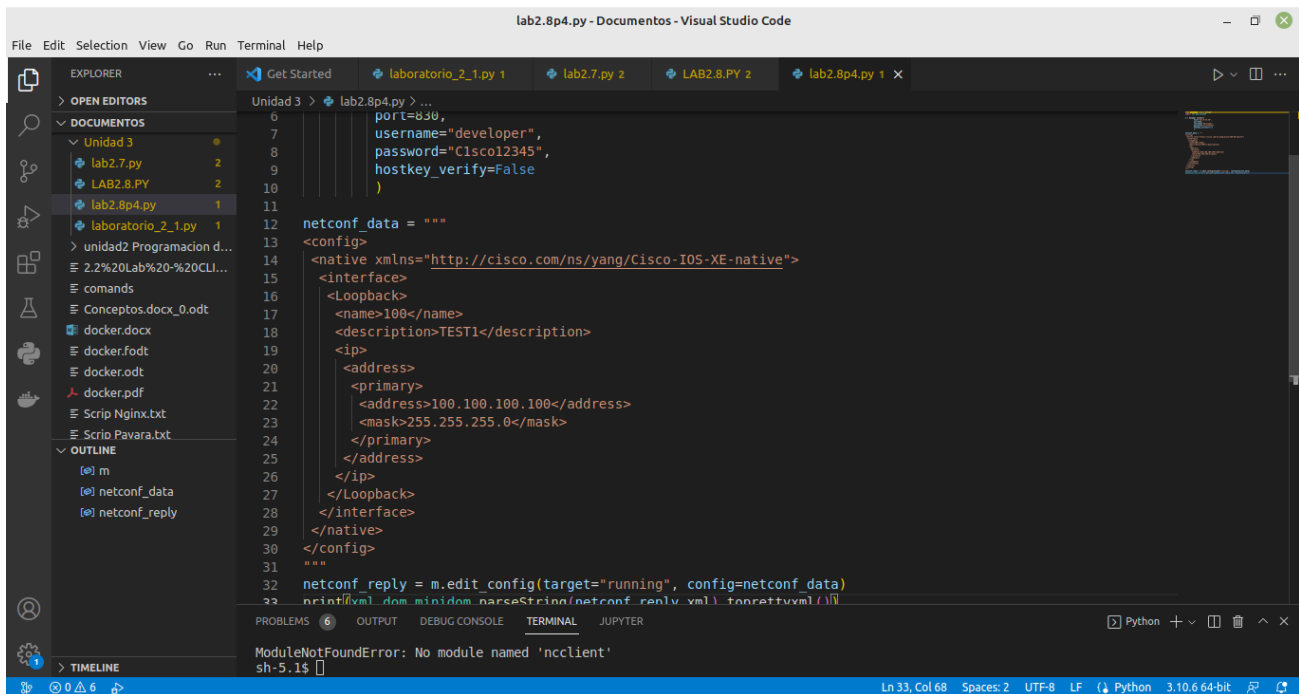
- o. Add the new loopback 100 interface by editing the existing device configuration using the `"edit_config()"` function:

```
netconf_reply = m.edit_config(target="running", config=netconf_data)
```

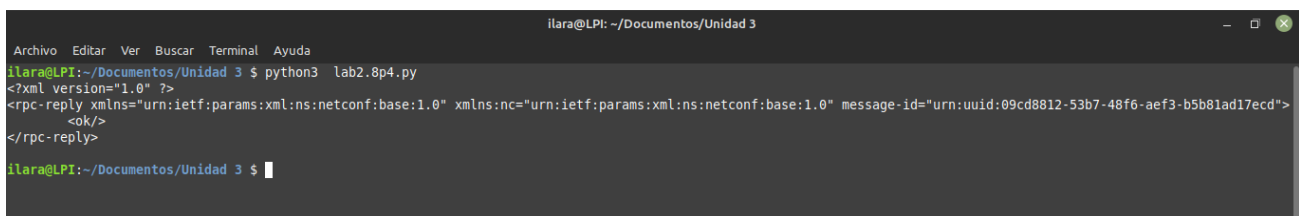
Lab – NETCONF w/Python: Device Configuration

```
print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

- p. 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.
- q. Execute the Python script and explore the output
- r. 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.



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



La respuesta que nos 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.

- a. 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.

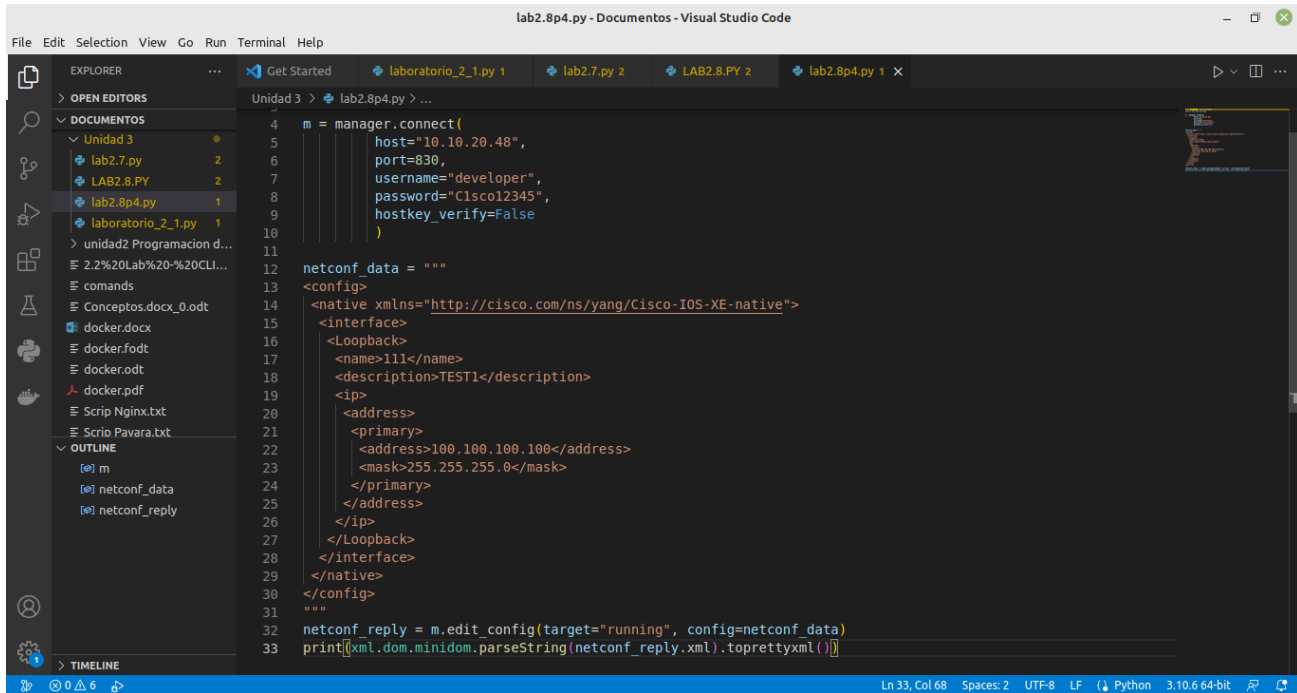
The device has refused one or more configuration settings. With NETCONF, thanks to the transactional behavior, no partial configuration change has been applied but the whole transaction was canceled.

- e. After executing the Python script, verify that no configuration changes, not even partial, have been applied.

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

Lab – NETCONF w/Python: Device Configuration

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 creado por tener un error.



```
lab2.8p4.py - Documentos - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
  OPEN EDITORS
  DOCUMENTOS
    Unidad 3
    lab2.7.py 2
    LAB2.8.PY 2
    lab2.8p4.py 1
    laboratorio_2_1.py 1
  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 Pavara.txt
  OUTLINE
    m
    netconf_data
    netconf_reply
  TIMELINE
  Unidad 3 > lab2.8p4.py > ...
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()]])
```



```
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.

Práctica de laboratorio, aprendimos a utilizar el ncclient de NETCONF para recuperar la configuración del dispositivo y actualizar y crear una nueva configuración de interfaz. También aprenderá por qué el soporte transaccional de NETCONF es importante para obtener cambios de red consistentes.

El módulo ncclient proporciona una clase de "administrador" con la función "conectar ()" para configurar la conexión NETCONF remota. Después de una conexión exitosa, el objeto devuelto representa la conexión NETCONF al dispositivo remoto. Después de una conexión NETCONF exitosa, use la función "get_config()" del objeto de sesión NETCONF "m" para recuperar e imprimir la configuración en ejecución del dispositivo. La función

`get_config()` espera un parámetro de cadena de "origen" que define el almacén de datos NETCONF de origen.

El módulo "xml.dom.minidom" se puede usar para embellecer la salida con la función `toprettyxml()`.

Se Creo la siguiente variable `netconf_filter` que contenga un elemento de filtro XML NETCONF diseñado para recuperar solo datos definidos por el modelo YANG nativo de Cisco IOS XE

Creamos una interfaz loopback correcta como lo hicimos con el modulo `netmiko` pero ahora con `yang` y sus librerías.

Creamos una que estaba en conflicto por que tenía mal la dirección por ello nos narco error.