



Área de TSU Infraestructura de  
Redes **Grupo:** GIR0441 **Materia:**  
Programación de redes

**Unidad:**

III programación de redes

**Actividad:**

2.8 Laboratorio-Netconf w-Python: Configuración del  
dispositivo

**Profesor:**

Ing. Gabriel Barrón

**No. Control:**

1221100727

**Alumna:**

Sandra Dania Gonzalez Manzano

**Lugar y Fecha:**

Dolores Hidalgo C.I.N a diciembre del 2022

## Investigación

Durante la practica también realice una reservación en sandbox para la conexión VPN como lo explicaba en el laboratorio 2.1

Ahora la practica se realizará nuevamente con NETCONF w/Python: Configuración del dispositivo,

Con este aprenderé un poco mas del ncclient de NETCONF para recuperar la configuración del dispositivo y actualizar y crear una nueva configuración de interfaz. El nsclient.exe es un archivo ejecutable en el disco duro de tu ordenador. El archivo contiene un código máquina. Si inicia el software Netscape Internet Service en tu PC, el comando que contiene nsclient.exe se ejecutará en tu PC. Para este fin, el archivo se carga en la memoria principal (RAM) y funciona ahí como un proceso de Netscape Internet Service (también denominado una tarea). La mayoría de los procesos no pertenecientes al sistema que se ejecutan pueden detenerse ya que no están involucrados en la ejecución de su sistema operativo. nsclient.exe es utilizado por 'Netscape Internet Service'. Esta aplicación ha sido creada por 'Netscape Communications Corporation'.

## Practica

En la parte 1 de recuperar la configuración de ejecución existente de la maquina virtual XE de iOS donde utilizare el modulo ncclient para recuperar la configuración en ejecución del dispositivo. Los datos se devuelven en formato XML.

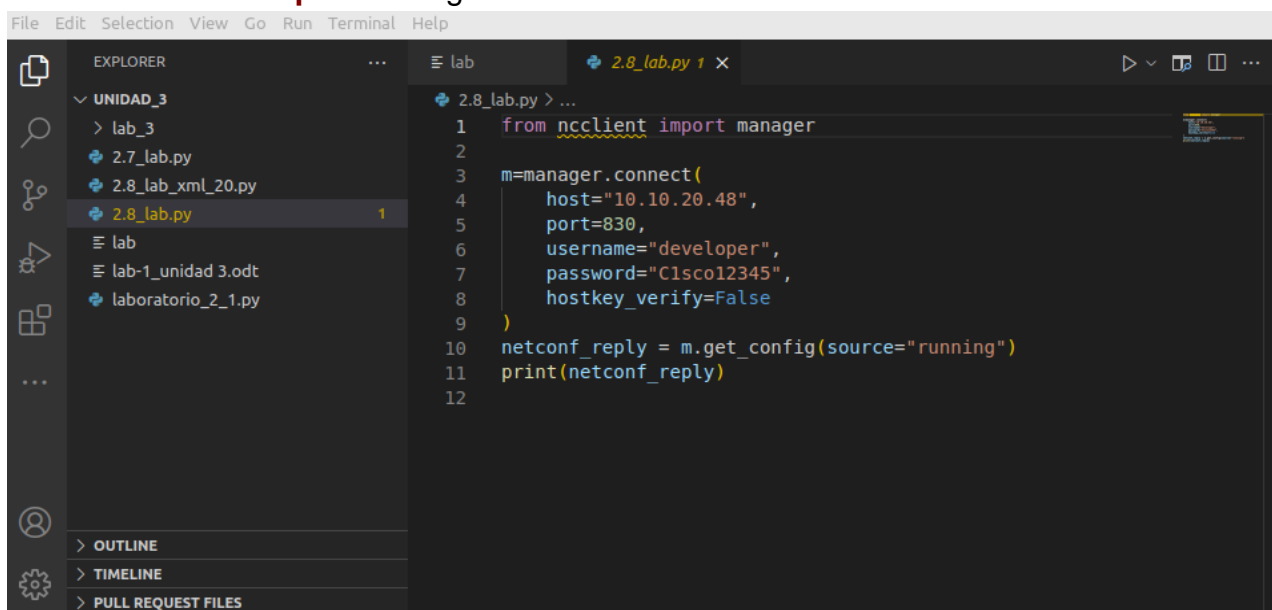
Dando pauta al primer paso donde se utilice ncclient para recuperar la configuración en ejecución del dispositivo.

Después de una conexión exitosa, el objeto devuelto representa la conexión NETCONF al dispositivo remoto.

En Python IDLE, cree un nuevo archivo de script de Python:

En el nuevo editor de archivos de script de Python, importe la clase "manager" desde el módulo ncclient:

**de NCCLIENT import manager**

A screenshot of a Python IDE window. The title bar shows 'File Edit Selection View Go Run Terminal Help'. The Explorer pane on the left shows a project structure with 'UNIDAD\_3' containing 'lab\_3', which includes files '2.7\_lab.py', '2.8\_lab\_xml\_20.py', '2.8\_lab.py' (selected), 'lab', 'lab-1\_unidad 3.odt', and 'laboratorio\_2\_1.py'. The main editor shows a Python script in '2.8\_lab.py' with the following code:

```
1 from ncclient import manager
2
3 m=manager.connect(
4     host="10.10.20.48",
5     port=830,
6     username="developer",
7     password="Cisco12345",
8     hostkey_verify=False
9 )
10 netconf_reply = m.get_config(source="running")
11 print(netconf_reply)
12
```

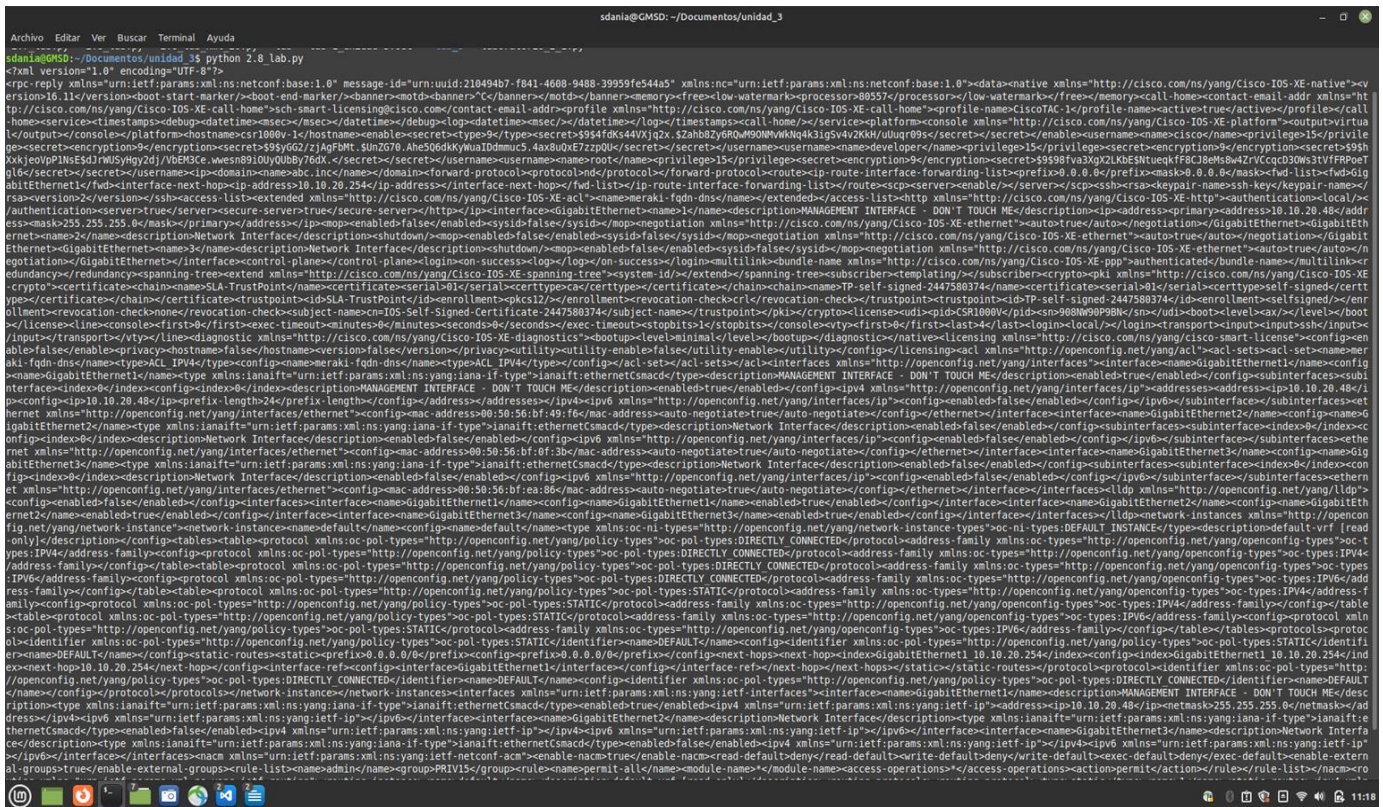
Los parámetros de la función manager.connect() son:

- **host** : esta es la dirección (host o IP) del dispositivo remoto (ajuste la dirección IP para que coincida con la dirección actual del router).
- **port** – Este es el puerto remoto del servicio SSH.
- **nombre** de usuario: este es el nombre de usuario SSH remoto (en este laboratorio, use "cisco" porque esa está configurada en la máquina virtual XE de iOS).
- **contraseña** : esta es la contraseña SSH remota (en este laboratorio, use "cisco123!" porque se configuró en la máquina virtual XE de iOS).

- **hostkey\_verify** – Utilice esto para verificar la huella digital SSH (en este laboratorio, es seguro establecer en False; sin embargo, en entornos de producción siempre debe verificar las huellas digitales SSH).

Con la función `manager.connect()`, configure un objeto de conexión m al dispositivo IOS

Ejecutando el script de Python



Y después del paso 2 Utilizando CodeBeautify.com para evaluar la respuesta

Code Beautify mantiene un sitio web para ver el código en un formato más legible para los humanos. La dirección URL del visor XML es <https://codebeautify.org/xmlviewer>

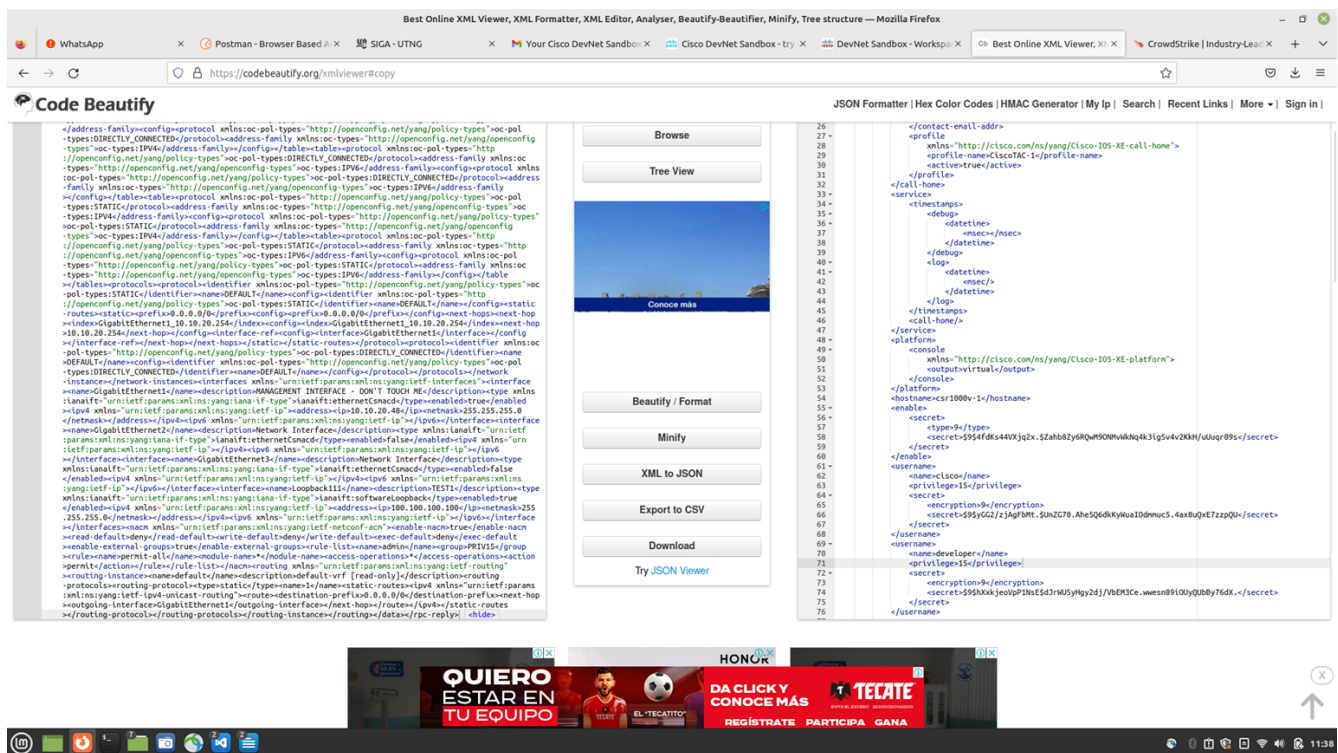
- Copie el XML de IDLE a XML Viewer.
- Haga clic en **Vista de árbol** o **Embellecer / Formato** para representar la salida XML sin procesar en un formato más legible por humanos.
- Para simplificar la vista, cierre los elementos XML que se encuentran bajo la estructura `rpc-reply/data`.



d. Tenga en cuenta que el elemento `rpc-reply/data/native` abierto contiene un atributo `xmlns` que apunta al modelo YANG "Cisco-IOS-XE-native ". Eso significa que esta parte de la configuración es Cisco Native para IOS XE.

e. También tenga en cuenta que hay dos elementos de "interfaces". El que tiene `xmlns` apunta al modelo YANG "http://openconfig.net/yang/interfaces", mientras que el otro apunta al modelo YANG "ietf-interfaces".

Ambos se utilizan para describir la configuración de las interfaces. La diferencia es que el modelo YANG `openconfig.net` admite subinterfaces, mientras que el modelo YANG de interfaces `ietf` no lo hace.



El paso 3 Utilice la función `toprettyxml` para embellecer la salida

Python ha incorporado soporte para trabajar con archivos XML. El módulo `"xml.dom.minidom"` se puede utilizar para embellecer la salida con la función `toprettyxml()`.

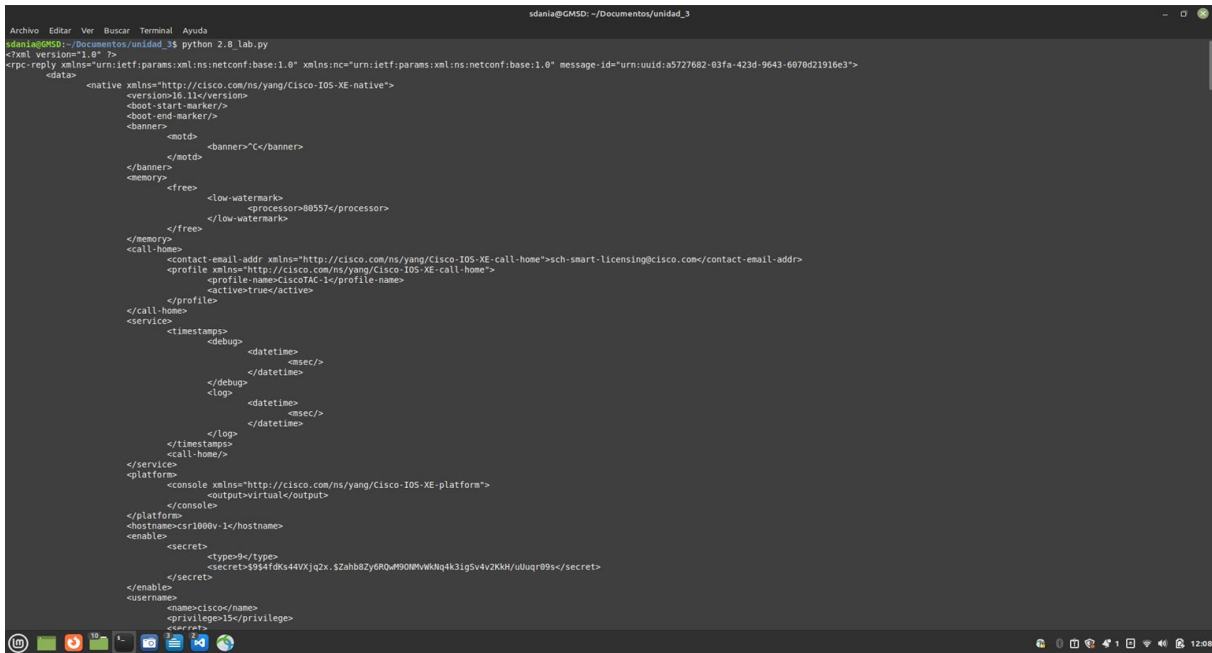
f. Importe el módulo `"xml.dom.minidom"`:

g. Importación XML. Inicio. `minidom`

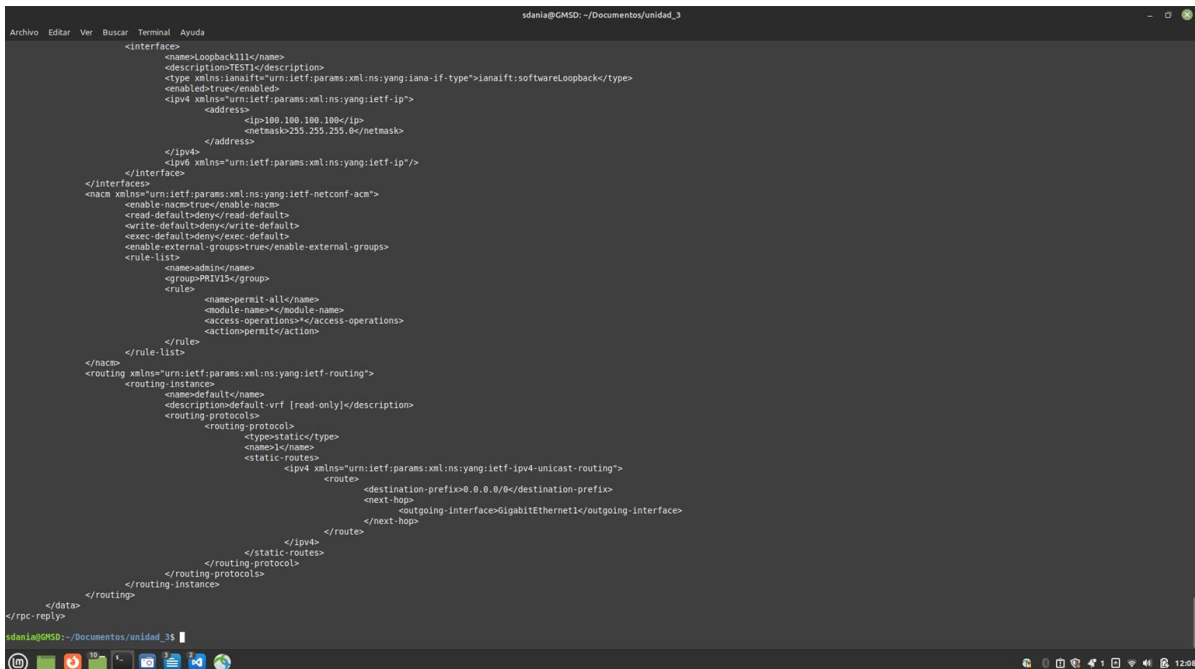
- h. Reemplace la función de impresión simple "print( netconf\_reply )" con una versión que imprima la salida XML embellecida:

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

- i. Ejecute el script de Python actualizado y explore la salida.



```
sdania@GMSD: ~/Documentos/unidad_3
sdania@GMSD:~/Documentos/unidad_3 python 2.8 lab.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:a5727682-03fa-423d-9643-6070d21916e3">
  <data>
    <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
      <version10.1i/version>
        <boot-start-marker/>
        <boot-end-marker/>
        <banner>
          <motd>
            <banners/C</banners>
          </motd>
        </banner>
        <memory>
          <free>
            <low-watermark>
              <processor>08557</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>CiscoIAC-1</profile-name>
            <active>true</active>
          </profile>
        </call-home>
        <service>
          <timestamps>
            <debug>
              <datetime>
                <sec/>
              </datetime>
            </debug>
            <log>
              <datetime>
                <sec/>
              </datetime>
            </log>
          </timestamps>
          <call-home/>
        </service>
        <platform>
          <console xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
            <output>virtual</output>
          </console>
        </platform>
        <hostname>cs1000v-1</hostname>
        <enable>
          <secret>
            <type>P</type>
            <secret>$9s4dK44Vxj2x.$Zahb2Y6RQwM90MwWkMq4k3ig5v4v2K5H/uduq99s</secret>
          </secret>
        </enable>
        <username>
          <name>Cisco</name>
          <privilege>15</privilege>
          <secret/>
        </username>
      </native>
    </data>
  </rpc-reply>
```



```
sdania@GMSD: ~/Documentos/unidad_3
sdania@GMSD:~/Documentos/unidad_3 python 2.8 lab.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:a5727682-03fa-423d-9643-6070d21916e3">
  <data>
    <interface>
      <name>Loopback11</name>
      <description>TEST1</description>
      <type xmlns:ianaif="urn:ietf:params:xml:ns:iana-if-type">ianaif:softwareLoopback</type>
      <enabled>true</enabled>
      <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
        <address>
          <ip>100.100.100.100</ip>
          <netmask>255.255.255.0</netmask>
        </address>
      </ipv4>
    </interface>
  </interface>
  <nacos xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm">
    <enable>true</enable>
    <read-default>deny</read-default>
    <write-default>deny</write-default>
    <exec-default>deny</exec-default>
    <enable-external-groups>true</enable-external-groups>
    <rule-list>
      <rule>
        <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>
    </nacos>
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <routing-instance>
      <name>default</name>
      <description>default-vrf [read-only]</description>
      <routing-protocol>
        <static-routes>
          <route>
            <type>static</type>
            <name>1</name>
            <destination-prefix>0.0.0.0/0</destination-prefix>
            <next-hop>
              <outgoing-interface>GigabitEthernet1</outgoing-interface>
            </next-hop>
          </route>
        </static-routes>
      </routing-protocol>
    </routing-instance>
  </routing>
</data>
</rpc-reply>
```

Utilice filtros para recuperar una configuración definida por un modelo YANG específico.

NETCONF tiene soporte para devolver solo datos que están definidos en un elemento de filtro.

Cree la siguiente variable `netconf_filter` que contiene un elemento de filtro XML NETCONF que está diseñado para recuperar sólo los datos definidos por el modelo Cisco IOS XE Native YANG:

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

Incluya la variable `netconf_filter` en la llamada `get_config()` usando el parámetro "filter":

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

```
2.8_lab.py > ...
1  from ncclient import manager
2  import xml.dom.minidom
3
4  m=manager.connect(
5      host="10.10.20.48",
6      port=830,
7      username="developer",
8      password="C1sco12345",
9      hostkey_verify=False
10 )
11 netconf_filter = """
12 <filtro>
13 |   <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native" />
14 </filtro>
15 """
16
17 netconf_reply = m.get_config(source="running", filter=netconf_filter)
18 print(xml.dom.minidom.parseString(netconf_reply.xml).toprettyxml())
```

Ejecute el script de Python actualizado

```
sdan@sdn:~/documentos/unidad_3$ python 2.8_lab.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:328e350b-96ec-490f-aa5-5de6aadc820">
  <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><?xml:space="preserve"><![CDATA[
          </banner>
        </motd>
      </banner>
      <memory>
        <low-watermark>
          <processor>805574</processor>
        </low-watermark>
      </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>CiscoIAC-1</profile-name>
          <active>true</active>
        </profile>
      </call-home>
      <service>
        <timestamp>
          <debug>
            <datetime>
              <sec/>
            </datetime>
          </debug>
          <log>
            <datetime>
              <sec/>
            </datetime>
          </log>
        </timestamp>
        <call-home/>
      </service>
      <platform>
        <console xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
          <output>v17</output>
        </console>
      </platform>
      <hostname>csr1000v-1</hostname>
      <enable>
        <secret>
          <type>secret</type>
          <secret>$9$4f0Ks44X1q2v.5Zeh82y6R0w4M00W4W4k431g54v2K3H/ubag/09s</secret>
        </secret>
      </enable>
    </native>
  </data>
</rpc-reply>
```

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
sdania@CMSD: ~/Documentos/unidad_3

    </pkcs12/>
    </enrollment>
    <revocation-check>url</revocation-check>
  </trustpoint>
  <trustpoint>
    <id>TP-self-signed-2447580374</id>
    <enrollment>
      <selfsigned/>
    </enrollment>
    <revocation-check>none</revocation-check>
    <subject-name>cn=IOS-Self-Signed-Certificate-2447580374</subject-name>
  </trustpoint>
</crypto>
<license>
  <udi>
    <pid>CSR1000V</pid>
    <sn>908NW90P90K</sn>
  </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>

sdania@CMSD:~/Documentos/unidad_3
```

Cree un nuevo archivo de script de Python.

- a. En IDLE, cree un nuevo archivo de script de Python.
- b. Importe los módulos necesarios y configure la sesión de NETCONF:

**Desde NCCLIENT Import Manager**  
Importación XML. Inicio. minidom

```
m = gerente.conectar(
    host="192.168.56.101",
    puerto = 830,
    username="cisco",
    password="cisco123!",
    hostkey_verify=False
)
```

Cambie el nombre de host.

- j. Para actualizar una configuración existente en la configuración, puede extraer la ubicación de la configuración recuperada en el paso 1.
- k. La actualización de configuración siempre está encerrada en un elemento XML "config". Este elemento incluye un árbol de elementos XML que requieren actualizaciones.
- l. Cree una variable **netconf\_data** que contenga una actualización de configuración para el elemento hostname como se define en el modelo Cisco IOS XE Native YANG:  
netconf\_data = ""



```

<config>
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
<nombre de host>NEWHOSTNAME</nombre de host>
  </nativo>
</config>
"""

```

m. Edite la configuración del dispositivo existente con la función "edit\_config()" del objetivo de sesión "m" de NETCONF. La función edit\_config() espera dos parámetros:

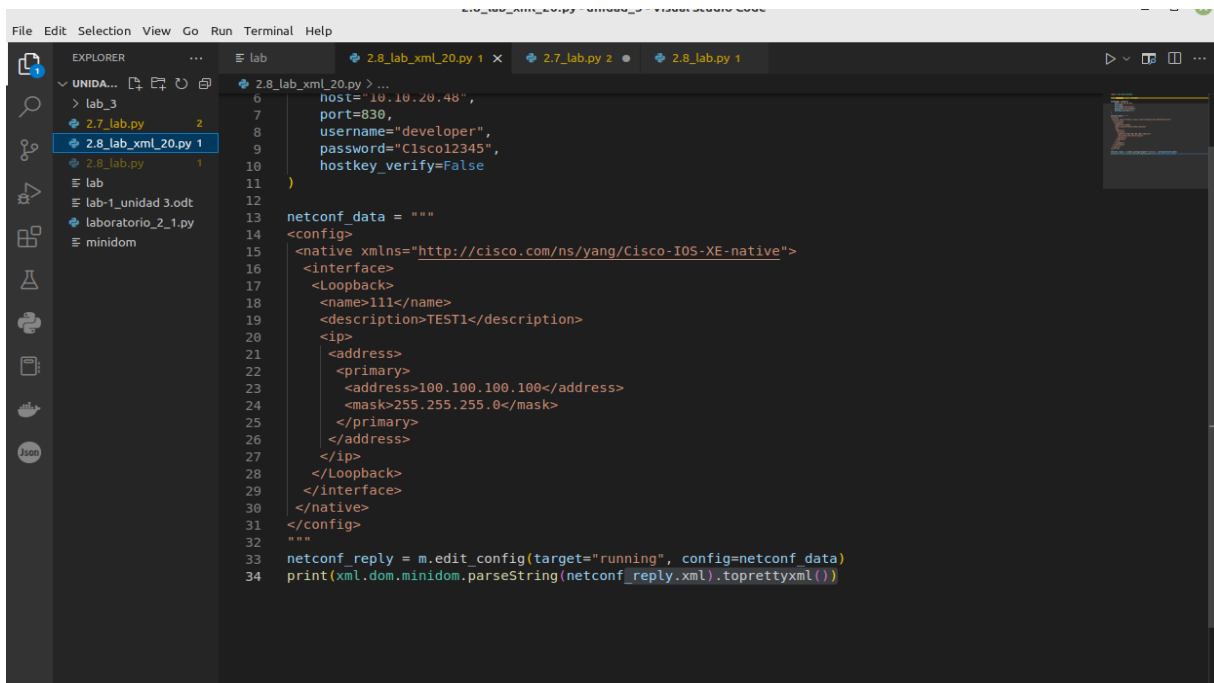
- **target** – Este es el almacén de datos netconf de destino que se actualizará.
- **config** – Esta es la actualización de configuración.

La función edit\_config() devuelve un objeto XML que contiene información sobre el éxito del cambio. Después de editar la configuración, imprima el valor devuelto:

```

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

```



Ejecute el script de Python y explore la salida. Después de ejecutar el script de Python, si la respuesta contenía el elemento <ok/>, verifique si se ha cambiado el nombre de host actual conectándose a la consola de la máquina virtual XE de iOS.

```

sdania@GMSD:~/Documentos/unidad_3$ ls
2.7 lab.py 2.8 lab.py 2.8 lab xml 20.py lab 'lab-1 unidad 3.odt' lab_3 laboratorio_2_1.py minidom
sdania@GMSD:~/Documentos/unidad_3$ python 2.8_lab_xml_20.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:b71340f1-2876-4903-b577-0e68482f83ea">
  <ok/>
</rpc-reply>
sdania@GMSD:~/Documentos/unidad_3$

```

## Conclusión

En el laboratorio 2.8 donde se a reforzado los protocolos que se han investigado durante los demás laboratorios donde se a investigado y practicado netconf ncclient, yang y la conexión en la sandbox.

Dando seguimiento a estos protocolos, librería y conexión de sandbox, practicando los comandos de configuración y darle formato a la salida que nos dio, donde embellecimos en formato del contenido además de que se utilizaron comandos para enviar configuraciones e loopback.