UNIVERSIDAD TECNOLÓGICA DEL NORTE DE GUANAJUATO

Tecnologías de la Información y Comunicación
Programa educativo:

TSU en Infraestructura de Redes Digitales
Área académica:

Programación de Redes
Asignatura:

Unidad III: Programación de Redes **Grupo:** GIR0441

CLI Automation with Python using netmiko
Laboratorio 1:

Venado Soria German Emiliano
Alumno:

Gabriel Barrón Rodríguez
Docente:

Dolores Hidalgo, C.I.N., Gto., Miércoles 14 de Diciembre de 2022
Lugar y fecha:

Lab – CLI Automation with Python using netmiko

Objectives

Part 1: Install the netmiko Python module

Part 2: Connect to IOS XE's SSH service using netmiko

Part 3: Use netmiko to gather information from the device

Part 4: Use netmiko to alter configuration on the device

Background / Scenario

For simple network automation using a remote telnet or ssh based command line, network administrators have been using various screen scraping techniques for a long period of time. Initially the "expect" based scripts we utilized to automate entering commands when a specific expected string appeared on the command line. With the evolution of the Python language, the netmiko Python module has emerged as an open source project hosted and maintained on GitHub.com that provides a simple network automation interface using similar techniques like the "expect" based scripts.

In this lab activity, you will identify the potential but also the limitations of using netmiko to transport CLI commands for network automation.

Required Resources

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

Instructions

Part 1: Install the netmiko Python module

In this part, you will install netmiko module into your Python environment. Netmiko is a python module that simplifies ssh CLI connection to network devices. It has built in functionality to identify to execute "exec mode" commands, as well as apply new commands in the running configuration.

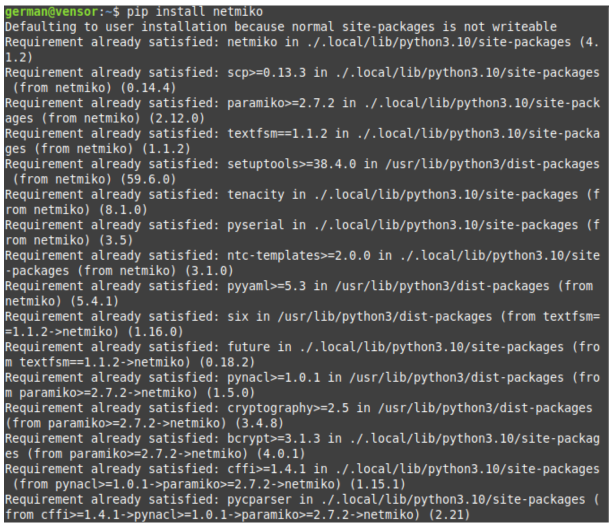
Explore the netmiko module on the project GitHub repository: https://github.com/ktbyers/netmiko

Step 1: Use pip to install netmiko.

a. Start a new Windows command prompt (`cmd`).

b. Install netmiko using pip in the Windows command prompt:

```
pip install netmiko
```

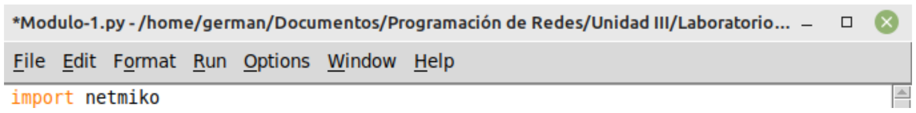


```
german@vensor:~$ pip install netmiko
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: netmiko in ./.local/lib/python3.10/site-packages (4.
1.2)
Requirement already satisfied: scp>=0.13.3 in ./.local/lib/python3.10/site-packages
 (from netmiko) (0.14.4)
Requirement already satisfied: paramiko>=2.7.2 in ./.local/lib/python3.10/site-pack
ages (from netmiko) (2.12.0)
Requirement already satisfied: textfsm==1.1.2 in ./.local/lib/python3.10/site-packa
ges (from netmiko) (1.1.2)
Requirement already satisfied: setuptools>=38.4.0 in /usr/lib/python3/dist-packages
 (from netmiko) (59.6.0)
Requirement already satisfied: tenacity in ./.local/lib/python3.10/site-packages (f
rom netmiko) (8.1.0)
Requirement already satisfied: pyserial in ./.local/lib/python3.10/site-packages (f
rom netmiko) (3.5)
Requirement already satisfied: ntc-templates>=2.0.0 in ./.local/lib/python3.10/site
-packages (from netmiko) (3.1.0)
Requirement already satisfied: pyyaml>=5.3 in /usr/lib/python3/dist-packages (from
netmiko) (5.4.1)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from textfsm=
=1.1.2->netmiko) (1.16.0)
Requirement already satisfied: future in ./.local/lib/python3.10/site-packages (fro
m textfsm==1.1.2->netmiko) (0.18.2)
Requirement already satisfied: pynacl>=1.0.1 in /usr/lib/python3/dist-packages (fro
m paramiko>=2.7.2->netmiko) (1.5.0)
Requirement already satisfied: cryptography>=2.5 in /usr/lib/python3/dist-packages
(from paramiko>=2.7.2->netmiko) (3.4.8)
Requirement already satisfied: bcrypt>=3.1.3 in ./.local/lib/python3.10/site-packag
es (from paramiko>=2.7.2->netmiko) (4.0.1)
Requirement already satisfied: cffi>=1.4.1 in ./.local/lib/python3.10/site-packages
 (from pynacl>=1.0.1->paramiko>=2.7.2->netmiko) (1.15.1)
Requirement already satisfied: pycparser in ./.local/lib/python3.10/site-packages (
from cffi>=1.4.1->pynacl>=1.0.1->paramiko>=2.7.2->netmiko) (2.21)
```

Con el comando proporcionado anteriormente también pude instalar netmiko en el sistema operativo Linux.

c. Verify that netmiko has been successfully installed. Start Python IDLE and in the interactive shell try to import the netmiko module:

```
import netmiko
```


*Modulo-1.py - /home/german/Documentos/Programación de Redes/Unidad III/Laboratorio...
File Edit Format Run Options Window Help

```
import netmiko
```

Part 2: Connect to IOS XE's SSH service using netmiko

Connect to IOS XE's SSH service using netmiko.

The netmiko module provides a "`ConnectHandler()`" function to setup the remote ssh connection. After a successful connection, the returned object represents the ssh cli connection to the remote device.

a. In Python IDLE, create a new Python script file:

b. In the new Python script file editor, import the "`ConnectHandler()`" function from the netmiko module:

```
from netmiko import ConnectHandler
```

En la captura anterior se solicita importar el módulo netmiko para trabajar con algunas funciones que integra. Cabe mencionar que me falto agregar en la captura la importación de la función de ConnectHandler, pero es el mismo comando que se Brinda en este paso, el cual será el encargado de adquirir los datos importantes como host, Puerto, usuario y contraseña, para que pueda hacerse la conexión remota.

c. Setup a `sshCli` connection object using the `ConnectHandler()` function to the IOS XE device.

The parameters of the `ConnectHandler()` function are:

- `device_type` – identifies the remote device type
- `host` – the address (host or IP) of the remote device (adjust the IP address "192.168.56.101" to match your router's current address)

- `port` – the remote port of the ssh service
- `username` – remote ssh username (in this lab “developer” for that was setup in the IOS XE VM)
- `password` – remote ssh password (in this lab “C1sco12345” for that was setup in the IOS XE VM)

```
sshCli = ConnectHandler (
    device_type = 'cisco_ios',
    host = '10.10.20.48',
    port = 22,
    username = 'developer',
    password = 'C1sco12345'
    )
```

En esta siguiente captura se aprecian los parámetros que debe ser ingresados en la función para generar la conexión remota hacia el Router que nos brindó el sandbox 19.9.3

Part 3: Use netmiko to gather information from the device

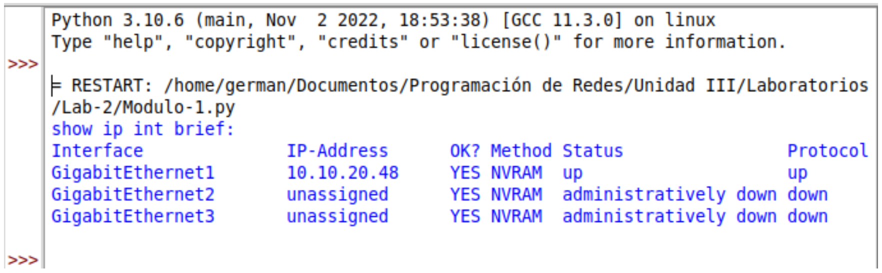
Send show commands and display the output

a. Using the `sshCli` object, returned by the `ConnectHandler()` function that represents the ssh cli remote session, send some “show” command and print the output. Use the `send_command()` function of the sshCli object with a string parameter that represents the command you wish to execute in the exec mode:

```
output = sshCli.send_command("show ip int brief")
print("show ip int brief:\n{}\n".format(output))
```

Con estos comandos podremos recopilar información del Router al que estamos teniendo acceso remotamente, ya que son respuestas de las funciones que utilizamos anteriormente.

b. Execute the Python script file to see the results.



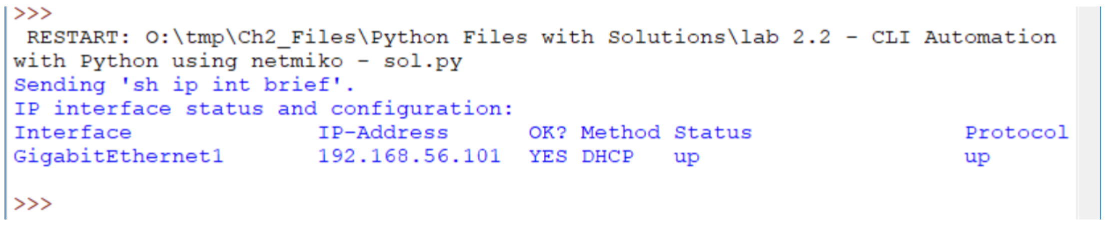
```
Python 3.10.6 (main, Nov  2 2022, 18:53:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: /home/german/Documentos/Programación de Redes/Unidad III/Laboratorios
/Lab-2/Modulo-1.py
show ip int brief:
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.10.20.48     YES NVRAM  up                    up
GigabitEthernet2       unassigned      YES NVRAM  administratively down down
GigabitEthernet3       unassigned      YES NVRAM  administratively down down

>>>
```

Como se puede observar, mi salida es similar a la que se proporciona en esta práctica, obviamente con pocos datos diferentes, como la dirección IP del dispositivo.

If you have not saved the script file yet, you will be prompted to save it before it is executed.

c. Verify the results:



```
>>>
 RESTART: O:\tmp\Ch2_Files\Python Files with Solutions\lab 2.2 - CLI Automation
with Python using netmiko - sol.py
Sending 'sh ip int brief'.
IP interface status and configuration:
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       192.168.56.101  YES DHCP   up                    up

>>>
```

d. Verify the data type of the "output" variable. How would you extract the IP address and the Interface Name into variables?

R= Tal vez esos datos los podría extraer con un tipo de filtro, donde indique los datos específicos esperados.

What if there were multiple interfaces?

R= Nada, simplemente todas aparecerían en la salida como fue que sucedió.

Part 4: Use netmiko to alter configuration on the device

In the following steps, you will alter the configuration of the device by creating new loopback interfaces.

Create a new loopback interface

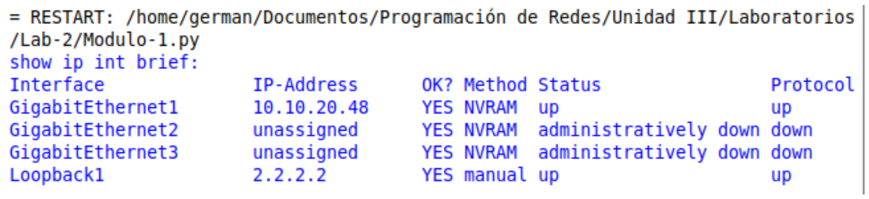
Using the `sshCli` object, returned by the `ConnectHandler()` function that represents the ssh cli remote session, send some configuration command and print the output. Use the `send_config_set()` function of the sshCli object with a list parameter including the configuration commands as strings you wish to execute in the exec mode:

```
config_commands = [
    'int loopback 1',
    'ip address 2.2.2.2 255.255.255.0',
    'description WHATEVER'
    ]

output = sshCli.send_config_set(config_commands)
```

En este punto se pide crear una interfaz loopback. Una interfaz loopback es una interfaz lógica que se asigna internamente en un Router, es útil para probar y administrar un dispositivo Cisco IOS, ya que asegura que por lo menos una interfaz esté siempre disponible. Con la función que se muestra en la captura se puede generar o crear la interfaz remotamente en el Router.

Execute the Python script file and verify the results



```
= RESTART: /home/german/Documentos/Programación de Redes/Unidad III/Laboratorios
/Lab-2/Modulo-1.py
show ip int brief:
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.10.20.48     YES NVRAM  up                    up
GigabitEthernet2       unassigned      YES NVRAM  administratively down down
GigabitEthernet3       unassigned      YES NVRAM  administratively down down
Loopback1              2.2.2.2         YES manual up                    up
```

Esta es la segunda salida, la cuál incluye la interfaz de loopback que fue creada anteriormente.

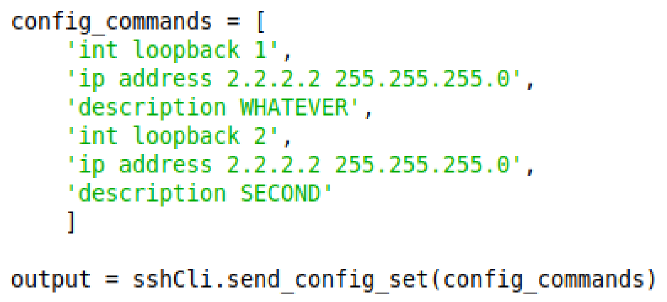
Why does the output from "show ip int brief" not include the "loopback1" interface?

R= Porque está mal estructurado el corchete de la configuración del dispositivo.

How to execute and display the output from the "show ip int brief" command after the loopback interfaces was created?

R= Simplemente acomodando el corchete al nivel de la sangría de la configuración.

Add code to create a new loopback interface (loopback2) with the same IP address as on the existing loopback interface, only with a different description.



```
config_commands = [
    'int loopback 1',
    'ip address 2.2.2.2 255.255.255.0',
    'description WHATEVER',
    'int loopback 2',
    'ip address 2.2.2.2 255.255.255.0',
    'description SECOND'
    ]

output = sshCli.send_config_set(config_commands)
```

Execute the Python script file and verify the results.

En esta ocasión no pude guardar la captura, pero la salida me marco un error.

Was the new loopback2 interface successfully created?

R= No es un éxito la creación de la segunda interfaz loopback, debido a que no se permite tener una dirección IP repetida.

Was the new configuration change accepted, partially accepted or rejected?

R= Se rechazó el cambio.

Investigaciones

- **¿Qué es netmiko?** Netmiko es una librería de redes multivendedores basada en Paramiko, que es una librería estándar para las conexiones SSH Python. Con Netmiko como base, se pueden realizar programas y scripts que faciliten y mejoren la administración de los equipos de redes.
- **¿Qué es SSH?**

  Es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente.

Conclusiones

De este segundo laboratorio me llevé nuevas experiencias con el módulo netmiko, comprendí que este modulo es bastante interesante para los usuarios que interactúan con las redes, pues todas sus funciones nos pueden ayudar a trabajar con otros dispositivos de manera remota, en esta ocasión trabajamos en base al protocolo SSH.