Lab – CLI Automation with Python using netmiko

1. 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

1. 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.

1. 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

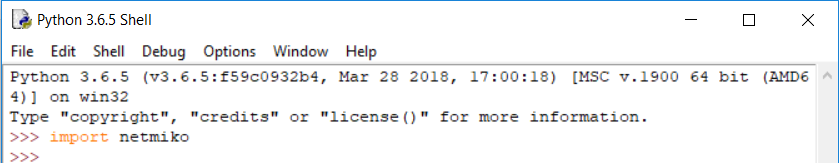
1. Install the netmiko Python module

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>

1. Use pip to install netmiko.
   * + 1. Start a new Windows command prompt (cmd).
       2. Install netmiko using pip in the Windows command prompt:  
            
          pip install netmiko
       3. Verify that netmiko has been successfully installed. Start Python IDLE and in the interactive shell try to import the netmiko module:

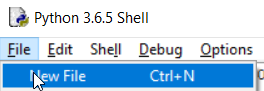
**import** netmiko



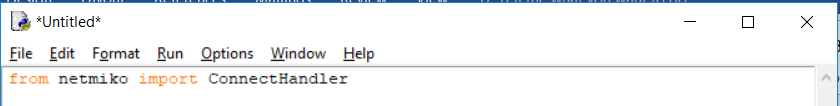
1. Connect to IOS XE’s SSH service using netmiko
   * 1. Connect to IOS XE’s SSH service using netmiko.

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.

* + - 1. In Python IDLE, create a new Python script file:



* + - 1. In the new Python script file editor, import the “ConnectHandler()” function from the netmiko module:  
           
         **from** netmiko **import** ConnectHandler



* + - 1. 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 “cisco” for that was setup in the IOS XE VM)
* password – remote ssh password (in this lab “cisco123!” for that was setup in the IOS XE VM)

1. Use netmiko to gather information from the device
   * 1. Send show commands and display the output
        1. 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:



* + - 1. Execute the Python script file to see the results: Take a screenshot of the result.

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

* + - 1. Verify the results:
      2. Verify the data type of the “output” variable. Would it be possible to parse the IP address and the Interface name and assigned to variables? Answer the question.  
         \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
         \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
         \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 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.

* + 1. Create a new loopback interface
       1. 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 = [

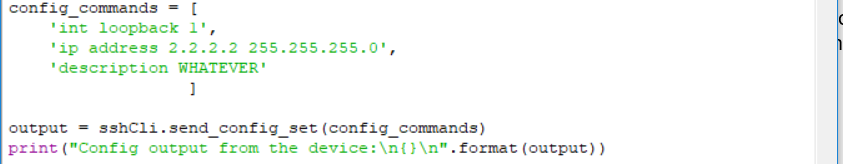
'int loopback 1',

'ip address 2.2.2.2 255.255.255.0',

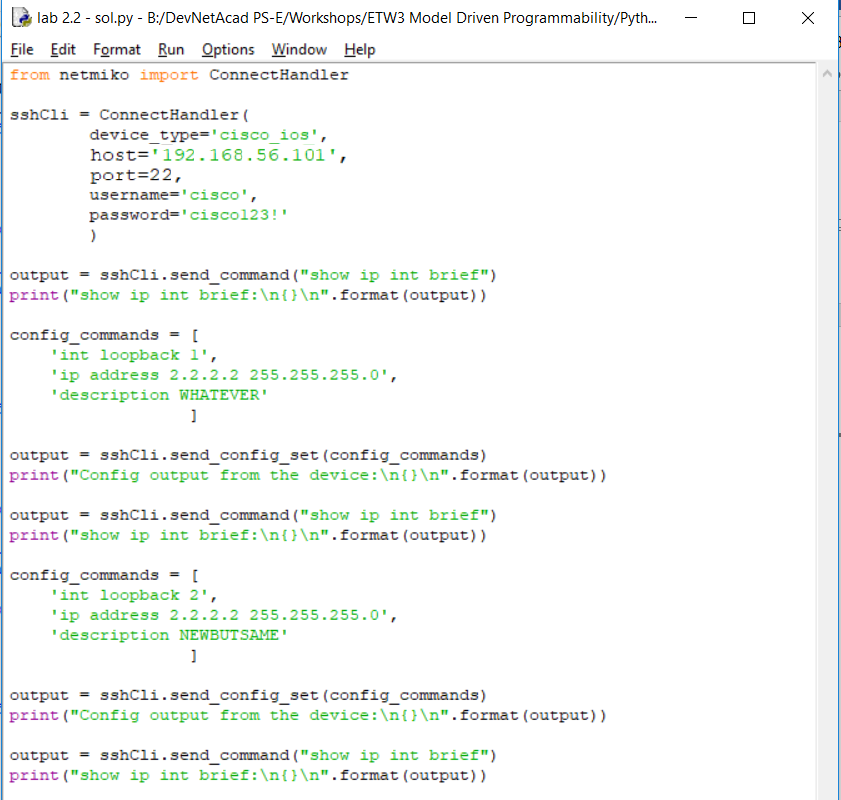
'description WHATEVER'

]

output = sshCli.send\_config\_set(config\_commands)



* + - 1. Execute the Python script file and verify the results: Take a screenshot of the result.
      2. Display the output from the “show ip int brief” command after the loopback interfaces was created? Take a screenshot of the result.
      3. 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.
      4. Execute the Python script file and verify the results: Take a screenshot of the result.



Paste Result Here

* + - 1. Was the new loopback2 interface successfully created? Answer the question.
      2. Was the new configuration change accepted, partially accepted or rejected? Answer the question.