Network Automation with Python

The networking industry is telling us all that we need to learn Python, programmability, and network automation. Using Python for Network Automation is a skill that all network engineers need to be developing. These labs will walk you through several different scenarios using Python to automate your network. This is a precursor to using Ansible, Nornir, and the DevNet course.

Equipment (Optional)

- One Linux PC
- One 2960 or later Switch running with K9 encryption (SSH capable)
- One 1841, 1941, 4000 series or later Router running with K9 encryption (SSH capable)
- Straight-thru and console cables as needed

Software

- Windows Subsystem for Linux enabled in Windows Programs and Features
- Windows Terminal app (install from Windows Store)
- Ubuntu (install from Windows Store)
- Python3 (latest version)
- PiP3 (latest version)
- Git (latest version)
- Netmiko (latest version)
- CISSHGO (Cisco router simulator)

Netmiko connection setup methods:

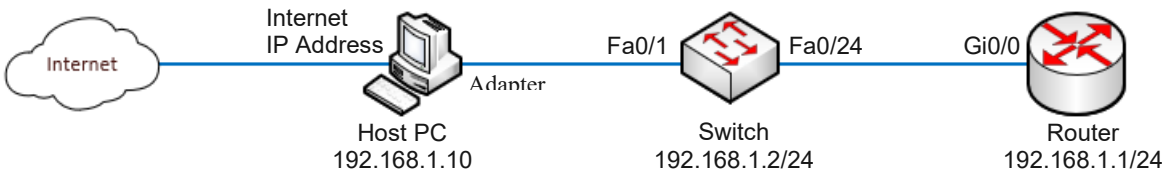
- **`device_type`** – Types of Cisco device software (cisco_ios, cisco_ios_telnet, cisco_asa, cisco_ios-xe, cisco_ios-xr, cisco_nx-os)
- **`ip`** – IP address of the Router or Switch (**`host`** is interchangeable)
- **`username`** – account name (if configured)
- **`password`** – account login password
- **`port`** – telnet: 23 or SSH: 22 (configured by default)
- **`secret`** – Privileged mode password (if configured)
- **`command`** – Configuration commands

Netmiko common connection methods:

- **`send_command()`** - Send command to a device and return output back (pattern based)
- **`send_config_set()`** - Send configuration commands to remote device
- **`send_config_from_file()`** - Send configuration commands loaded from a file
- **`enable()`** - Enter enable mode
- **`find_prompt()`** - Return the current router prompt
- **`save_config()`** - Save the running-config to the startup-config
- **`disconnect()`** - Close the connection
- **`send_command_expect()`** - Wait for a command to finish (timing based)
- **`send_command_timing()`** - Send commands to a device and return output back (timing based)
- **`commit()`** - Execute a commit action on IOS-XR
- **`check_config_mode()`** - Check config mode status
- **`check_enable_mode()`** - Check enable mode status
- **`exit_config_mode()`** - Exit Global Configuration

- **`exit_enable_mode()`** - Exit Privileged mode

Topology



Part 1: Basic Switch and Router Configuration (only complete Part 1 if you are using real equipment)

1. Configure the Host PC network adaptor with the information below:

   Host PC

   IP address: 192.168.1.10
   SM: 255.255.255.0
   DG: 192.168.1.1

2. From the Ubuntu terminal, check the IP configuration (**`ip address`**).
3. Each network device needs a basic configuration before starting to gain access over the network. Using a rollover cable and your preferred terminal app (PuTTY, HyperTerminal), configure a basic router and switch with SSH capabilities:
   Note: Only configure these settings (interfaces may vary).

   Router

```
enable
configure terminal
hostname R1
enable secret class
username admin password cisco
ip domain name netauto.com
crypto key generate rsa
1024
ip ssh version 2
interface GigabitEthernet0/0
ip address 192.168.1.1 255.255.255.0
no shutdown
exit
line con 0
login local
line vty 0
login local
transport input ssh
end
terminal length 0
copy run start
```

   Switch

```
enable
configure terminal
hostname S1
enable secret class
username admin password cisco
ip domain-name netauto.com
crypto key generate rsa
1024
ip ssh version 2
interface vlan 1
ip address 192.168.1.2 255.255.255.0
no shutdown
exit
ip default-gateway 192.168.1.1
line con 0
login local
line vty 0
login local
transport input ssh
end
terminal length 0
copy run start
```

4. Be sure you can ping between all devices. Troubleshoot as needed.
5. Add other devices as needed.

Part 2: Initial Setup and Cloning

1. Create a folder named **wsl** on your physical PC in your current account:
   C:\Users*Account_Name*\wsl
2. Open Windows Terminal and open an Ubuntu instance. You should see a prompt similar to this:
   ```
   cisco@Scott-HomePC:/mnt/c/Users/scott$
   ```
3. Change directories to the **wsl** you just created on Windows:
   cd wsl
   ```
   cisco@SCOTT-HOMEPC:/mnt/c/Users/scott/wsl$
   ```
4. Clone my CISSHGO repository to the **wsl** directory:
   git clone https://github.com/Scott4564/cisshgo_for_devnet.git
   Note: This creates a new folder on your physical machine at C:\Users*Account_Name*\wsl\cisshgo_for_devnet.
5. Clone my GitHub Programmability repository and then change directories into it:
   git clone https://github.com/Scott4564/programmability.git
   cd programmability
   Note: This creates a new folder on your physical machine at C:\Users*Account_Name*\wsl\programmability.
6. Start the CISSHGO server with 10 instances (50 is default):
   cd ..
   cd cisshgo_for_devnet
   go run cissh.go -listeners 10
   ```
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10009
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10002
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10003
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10000
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10004
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10005
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10001
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10006
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10007
   2022/04/19 19:09:43 Starting cissh.go ssh server on port :10008
   ```
   Note: The port numbers will be list in a random order.
7. Open a new Ubuntu tab in Windows Terminal.
8. Display the Ubuntu's IP address:
   ip address
   ```
   1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
       inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever
   2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
       link/ether 00:15:5d:88:44:9b brd ff:ff:ff:ff:ff:ff
       inet 172.22.87.158/20 brd 172.22.95.255 scope global eth0
          valid_lft forever preferred_lft forever
       inet6 fe80::215:5dff:fe88:449b/64 scope link
          valid_lft forever preferred_lft forever
   ```

9. Write down the ethernet's IP address and the port number you will use (assign every student a different port number). In my case the IP address is **172.22.87.128** and the CISSHGO server port is **10000**.

10. Test your SSH connection with username **cisco** and password **cisco**:

```
ssh cisco@172.22.87.158 -p 10000
The authenticity of host '[172.22.87.158]:10000 ([172.22.87.158]:10000)' can't be established.
RSA key fingerprint is SHA256:qht/wZ3hThPHI08DTc3aLhkBc+w6wKbO6Y3574OGcak.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[172.22.87.158]:10000' (RSA) to the list of known hosts.
cisco@172.22.87.158's password: cisco
R1>
```

Note: You could use a terminal emulator (PuTTY) to make the connection.

Part 3: Connect to a Router using Python and Netmiko

The best way to connect is with a SSH connection, which we will be using.
You can connect to a Cisco device using Python is via telnet, but for most environments, telnet is disabled since it's not secure.

The **Netmiko** library has made it easier to connect to network devices using Python & SSH. This library works across a broad variety of networking devices, including Cisco.

Note: Change the IP address, username, and password to what you configured in Part 1.

1. Run these commands from the **Python3** command prompt to show the interfaces on the Router:

```
python3
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> from netmiko import ConnectHandler
>>>
```

1 line of code

```
>>> connection = ConnectHandler(device_type='cisco_ios', ip='172.22.87.158',
port=10000, username='cisco', password='cisco')
>>>
>>> print(connection.send_command('show ip int brief'))
Interface                  IP-Address      OK? Method Status                Protocol
FastEthernet0/0            192.168.1.1     YES NVRAM  up                    up
FastEthernet0/1            192.168.12.1    YES NVRAM  up                    up
Serial0/0                  unassigned      YES NVRAM  administratively down down
Serial0/1                  unassigned      YES NVRAM  administratively down down
FastEthernet0/1/0          unassigned      YES NVRAM  down                  down
FastEthernet0/1/1          unassigned      YES NVRAM  down                  down
FastEthernet0/1/2          unassigned      YES unset  down                  down
FastEthernet0/1/3          unassigned      YES unset  down                  down
Loopback0                  1.1.1.1         YES NVRAM  up                    up
Vlan1                      unassigned      YES NVRAM  down                  down
>>> connection.disconnect()
>>>
>>> exit()
```

Part 4: Connect to a Switch or Router using Python from a Script

1. Open and type the following code in a text editor (Nano, VIM, VS Code) and save it as **show-int.py**:

```
cd /mnt/c/Users/Scott/wsl/programmability
sudo nano show-int.py
```

```
#!/usr/bin/python3

import netmiko

# Connect to a Cisco Router using SSH and run show commands
router = {
    'device_type': 'cisco_ios',
    'ip': '172.22.87.158'',
    'port': '10000',
    'username': 'cisco',
    'password': 'cisco',
    'secret': 'class',
}
# Establish a connection
connection = netmiko.ConnectHandler(**router)

# Show output
output = connection.send_command('show ip int brief')
device = connection.find_prompt()
print (device + '\n' + output)

# Disconnect
connection.disconnect()
```

2. Save and Exit.
3. Run the Python script and you should see an output of the IP interfaces on your Router or Switch.

```
python3 show-int.py
```

```
R1>
Interface              IP-Address      OK?   Method   Status                  Protocol
GigabitEthernet0/0     192.168.1.1     YES   manual   up                      up
GigabitEthernet0/1     unassigned      YES   unset    administratively down   down
S1>
Interface              IP-Address      OK?   Method   Status                  Protocol
Vlan1                  192.168.1.2     YES   manual   up                      up
FastEthernet0/1        unassigned      YES   unset    up                      up
<Output omitted>
FastEthernet0/24       unassigned      YES   unset    up                      up
GigabitEthernet0/1     unassigned      YES   unset    down                    down
GigabitEthernet0/2     unassigned      YES   unset    down                    down
```

4. Open the script again but this time redirect the results to a file named **show.txt**. Add this text below the print line:

```
# Output results to a file
with open("show.txt", "w") as o:
    o.write(device + '\n' + output + '\n')
```

Note: Use **"a"** to append, **"w"** to write, **"r"** to read.

5. Run the script again:

python3 show-int.py

6. Check the file was created:

ls

```
'Prerequisites for Network Automation Lab.pdf'   README.md   show-int.py   show.txt
```

7. Show the contents of the **show.txt** file:

cat show.txt

Note: You can also use **nano**, **vim**, **more**, **less**, **head**, or **tail** to view the contents of a file.

8. Close the CISSHGO server:

Ctrl + c

Close the tab.

Part 5: Upload Your Results Back to GitHub

1. Be sure you are in the **programmability** directory.
2. Initialize the repository

git init

3. Use your **FirstName_LastName** as the name for a new branch

git branch *FirstName_LastName*

4. Make this the active branch:

git checkout *FirstName_LastName*

5. Look at the status of the repository:

git status

6. Add everything in the directory to this branch

git add .

7. Commit changes using your First and Last name as the message for the commit

git commit -m "*FirstName_LastName*"

8. You may be asked to provide your username and email address, then commit again*:

git config --global user.email "*you@example.com*"
git config --global user.name "*Your Username*"
git commit -m "*FirstName_LastName*"

9. Upload the new branch to GitHub:

git push --set-upstream origin *FirstName_LastName*

*Note: Your instructor will provide you with the account information when you are ready to upload:

- Username: **CiscoProgrammability**
- Email: **cisco123.student@gmail.com**
- Personal Access Token: **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**