Network Automation with Python

The networking industry is telling us all that we need to learn Python, programmability, and network automation. But for most of us using the CLI nothing has changed, and we are thinking what is all the fuss about? Using Python for Network Automation is a skill that all network engineers need to be developing. This lab 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

- One Windows 10 or later PC with two network adapters (one Internet connections; one LAN connection)
- 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)
- Three Straight-thru cables
- One Rollover cable for initial configuration

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)
- Visual Studio Code (any text editor like Nano) to edit Python code
- Django web server (latest version)
- json2html

Netmiko connection setup methods:

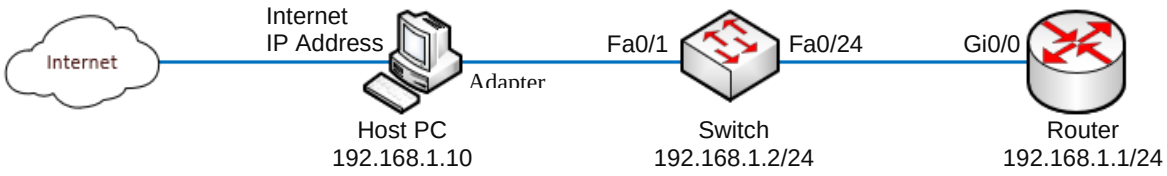
- **`device_type`** – Types of Cisco device software (cisco_ios, cisco_ios_telnet, cisco_asa, cisco_ios-xe, cisco_ios-xr, cisco_nx-os)
- **`ip`** – IP address of the Router or Switch (**`host`** is interchangeable)
- **`username`** – account name (if configured)
- **`password`** – account login password
- **`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

1. You should have two interfaces. One to the internet and one to the practice network. Configure the practice 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: Starting Python Files

1. Clone my GitHub repository and then change directories into it:

```
git clone https://github.com/Scott4564/python.git
cd python
```

Note: This creates a new folder on your physical machine at C:\Users*Account_Name*\python.

Part 3: Connect to a Switch or Router using Python and Netmiko

The simplest way to connect to a Cisco device using Python is via telnet, but for most environments, telnet is disabled. The best option is SSH, which we will use for the initial show and configuration commands.

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:

```
from netmiko import ConnectHandler
connection = ConnectHandler(device_type='cisco_ios', ip='192.168.1.1',
username='admin', password='cisco')
print(connection.send_command('show ip int brief'))
connection.disconnect()
```

(1 line of code)

2. Run these commands from the Python3 command prompt to show the interfaces on the Switch:

```
from netmiko import ConnectHandler
connection = ConnectHandler(device_type='cisco_ios', ip='192.168.1.2',
username='admin', password='cisco')
print(connection.send_command('show ip int brief'))
connection.disconnect()
exit()
```

(1 line of code)

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, VS Code) and save it as **show-int.py**:
Note: The Python3 "She-bang" **#!/usr/bin/python3** is always recommended but has been omitted in these examples.

```
import netmiko

# Connect to a Cisco Router using SSH and run show commands
router = {
    'device_type': 'cisco_ios',
    'ip': '192.168.1.1',
    'username': 'admin',
    '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()

# Connect to a Cisco Switch using SSH and run show commands
switch = {
    'device_type': 'cisco_ios',
    'ip': '192.168.1.2',
    'username': 'admin',
    'password': 'cisco',
    'secret': 'class',
}
# Establish a connection
connection = netmiko.ConnectHandler(**switch)

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

# Disconnect
connection.disconnect()
```

2. 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
```

Part 5: Configure an interface on a Cisco Router

1. Open and add the following code in a text editor (Nano, VS Code) and save it as **write-config.py**
   Note: Copy the Router code from show-int.py and add some configuration information so that your Python script can actually configure the router.

```
import netmiko

# Use Netmiko to connect to a Cisco Router using SSH and run simple commands
router = {
    'device_type': 'cisco_ios',
    'ip': '192.168.1.1',
    'username': 'admin',
    'password': 'cisco',
    'secret': 'class',
}
# Commands to configure a loopback
config_commands = ['interface loopback 0', 'ip address 1.1.1.1
255.255.255.255']

# Establish a connection
connection = netmiko.ConnectHandler(**router)

# Enter privileged mode
connection.enable()

# Run commands
config = connection.send_config_set(config_commands)
print (config)

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

# Disconnect
connection.disconnect()
```

2. Run the Python script and you should see an output of the IP interfaces on your Router.

```
python3 write-config.py
config term
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface loopback 0
R1(config-if)#ip address 1.1.1.1 255.255.255.255
R1(config-if)#no shutdown
R1(config-if)#end
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
Loopback0              1.1.1.1       YES   manual   up                      up
```

Part 6: Configure multiple devices at once with one command

1. Open and add the following code in a text editor (Nano, VS Code) and save it as **multiple-config.py**
2. Add the following code:

```
import netmiko

router = {
    'device_type': 'cisco_ios',
    'ip': '192.168.1.1',
    'username': 'admin',
    'password': 'cisco',
    'secret': 'class',
    'command': ['interface loopback 1', 'ip address 2.2.2.2 255.255.255.255',
                'interface loopback 2', 'ip address 3.3.3.3 255.255.255.255'],
    'output': 'show ip int brief'
}

switch = {
    'device_type': 'cisco_ios',
    'ip': '192.168.1.2',
    'username': 'admin',
    'password': 'cisco',
    'secret': 'class',
    'command': ['vlan 10', 'name TEACHERS', 'vlan 20', 'name STUDENTS',
                'vlan 30', 'name ADMINISTRATORS'],
    'output': 'show vlan brief'
}

# Create a loop to run commands
for device in (router, switch):
    # Retrieve the configuration commands and fill a variable
    command = device.pop('command')
    output = device.pop('output')
    connection = netmiko.ConnectHandler(**device)
    connection.enable()

    # Send the commands to the devices and view an output
    commands = connection.send_config_set(command)
    print (commands)
    outputs = connection.send_command(output)
    print (outputs)
    connection.disconnect()
```

3. Run the Python script and you should see an output on your Router or Switch.

python3 multiple-config.py

```
config term
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface loopback 1
R1(config-if)#ip address 2.2.2.2 255.255.255.255
R1(config-if)#no shutdown
R1(config-if)#interface loopback 2
R1(config-if)#ip address 3.3.3.3 255.255.255.255
R1(config-if)#no shutdown
R1(config-if)#end
R1#
Interface              IP-Address    OK?   Method   Status                    Protocol
GigabitEthernet0/0     192.168.1.1   YES   manual   up                        up
GigabitEthernet0/1     unassigned    YES   manual   administratively down    down
Loopback0              1.1.1.1       YES   manual   up                        up
Loopback1              2.2.2.2       YES   manual   up                        up
Loopback2              3.3.3.3       YES   manual   up                        up
config term
Enter configuration commands, one per line.  End with CNTL/Z.
S1(config)#vlan 10
S1(config-vlan)#name TEACHERS
S1(config-vlan)#vlan 20
S1(config-vlan)#name STUDENTS
S1(config-vlan)#vlan 30
S1(config-vlan)#name ADMINISTRATORS
S1(config-vlan)#end
S1#
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4, Fa0/5, Fa0/6,
                                                Fa0/7, Fa0/8, Fa0/9, Fa0/10, Fa0/11, Fa0/12,
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/16, Fa0/17,
                                                Fa0/18, Fa0/19, Fa0/20, Fa0/21, Fa0/22,
                                                Gi0/1, Gi0/2
10   TEACHERS                         active
20   STUDENTS                         active
30   ADMINISTRATORS                   active
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
```

Part 7: Configure Device from a File

1. If needed, clone your **router_config.txt** and **switch_config.txt** files to the Ubuntu subsystem:
   ls
   ```
   config.py  router_config.txt  show-config.py  show-int.py  switch_config.txt
   ```
2. Change the configs to match your device interfaces.
3. Open and add the following code in a text editor (Nano, VS Code) and save it as **config-file.py**
4. Add the following code:

```
import netmiko

router = {
    'device_type': 'cisco_ios',
    'ip': '192.168.1.1',
    'username': 'admin',
    'password': 'cisco',
    'secret': 'class',
    'command': 'router_config.txt',
    'output': 'show ip int brief'
}

switch = {
    'device_type': 'cisco_ios',
    'ip': '192.168.1.2',
    'username': 'admin',
    'password': 'cisco',
    'secret': 'class',
    'command': 'switch_config.txt',
    'output': 'show vlan brief'
}

# Create a loop to run commands
for device in (router, switch):
    # Retrieve the configuration commands and fill a variable
    command = device.pop('command')
    output = device.pop('output')
    connection = netmiko.ConnectHandler(**device)
    connection.enable()

    # Send the commands to the devices and view an output
    commands = connection.send_config_set(command)
    print (commands)
    outputs = connection.send_command(output)
    print (outputs)
    connection.disconnect()
```

5. Run the Python script and you should see an output on your Router or Switch.
   python3 config-file.py

Part 8: Creating a Webpage to View Interfaces

1. Install Django and json2html applications:
   pip3 install Django
   pip3 install json2html
2. From the Python directory, add a new project named **webint**:
   django-admin startproject webint
3. Change to the **webint** directory and install ntc-templates and set the path:
   cd webint
   git clone https://github.com/networktocode/ntc-templates.git
   export NET_TEXTFSM='./ntc-templates/ntc_templates/templates'
4. Migrate the settings:
   ls
   ```
   db.sqlite3  manage.py  ntc-templates  webint
   ```
   python3 manage.py migrate
5. Run the server and test the website:
   python3 manage.py runserver
   ```
   Watching for file changes with StatReloader
   Performing system checks...

   System check identified no issues (0 silenced).
   April 30, 2020 - 21:43:23
   Django version 3.0.5, using settings 'showint.settings'
   Starting development server at http://127.0.0.1:8000/
   Quit the server with CONTROL-C.
   ```
6. Open a browser and type the URL shown in the output (**127.0.0.1:8000/**).

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

7. Use **CTRL+C** in the command prompt to stop the server.
8. Change to the next level **webint** directory.
   cd webint
   ls
   __init__.py __pycache__ asgi.py settings.py urls.py wsgi.py

9. Open **urls.py** and append this code:
   sudo nano urls.py

```
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.output, name='output'),
]
```

10. Save and close the file
11. Create a new file named **views.py** and add this code:
    Note: You can reuse some of the code from show-int.py.
    sudo nano views.py

```
from django.http import HttpResponse
import netmiko
import json
from json2html import *
from datetime import datetime

def output(request):
    router = {
        'device_type': 'cisco_ios',
        'ip': '192.168.1.1',
        'username': 'admin',
        'password': 'cisco',
        'secret': 'class',
        'command': 'show ip int brief'
    }

    clicommand = router.pop('command')

    connection = netmiko.ConnectHandler(**router)

    connection.enable()

    output = connection.send_command(clicommand, use_textfsm=True)

    # Get device name
    device = connection.find_prompt()

    connection.disconnect()

    # Get the current time
    t = datetime.now()
```

```
        # Format the output as a readable HTML page
        show = json2html.convert(json = output)

        # Display on a webpage
        return HttpResponse('<center>' + str(t) + '<br><br><span style="font-
        size:20px;"><b>' + clicommand + ' for ' + str(device) +'</b></span><br>'
        + show)
```

1 line of code

12. Save and close the file.
13. Back up one level and run the server again:

```
cd ..
python3 manage.py runserver
```

14. Refresh the browser page 127.0.0.1:8000/
15. After approximately 30 seconds you should see the results load

2020-05-04 17:17:05.325548

show ip int brief for R1#

intf	ipaddr	status	proto
GigabitEthernet0/0	192.168.1.1	up	up
GigabitEthernet0/1	unassigned	administratively down	down
Loopback0	1.1.1.1	up	up
Loopback1	2.2.2.2	up	up
Loopback2	3.3.3.3	up	up
Loopback3	4.4.4.4	up	up

Note: You can change the show command and see different results.

Part 9: Backup the Configuration Files Using SSH

1. From the python folder in a terminal, open VS Code:
   Code .
2. Create a new text file named **hosts.txt** in the Python folder:
   Python > hosts.txt
3. Add the IP addresses of all the devices you want to get the configurations:
   192.168.1.1
   192.168.1.2
4. Save and close the file.
5. Create a new file named **credentials.py** in the Python folder:
   Python > credentials.py
6. Add this information code:

```
username = "admin"
password = "cisco"
secret = "class"
```

7. Save and close the file.
8. Create a new Python script named **backup-config-ssh.py**
   Python > backup-config.py
9. Add the following code:

```
#!/usr/bin/env python3

import credentials
import netmiko

# Retrieve the username, password, and secret
user = credentials.username
pass = credentials.password
sec = credentials.secret

# Open the hosts.txt file containing the device IPs
devices = open('hosts.txt')
# Create a connection string
for line in devices:
    connection = netmiko.ConnectHandler(device_type='cisco_ios', ip=line,
    username=user, password=pass, secret=sec)

    # Run commands
    connection.send_command('terminal length 0')
    config = connection.send_command('show run')
    device = connection.find_prompt()

    # Write config to file
    File_object = open(device + "-config.txt","w")
    File_object.writelines(config)
    connection.disconnect()
```

1 line of code

10. Save and run the file:
    Run > Run Without Debugging

Supplemental: Remote Through a Terminal Server Device Using Telnet

1 line of code

```
# Taken from Part 3
from netmiko import ConnectHandler
connection = ConnectHandler(device_type='cisco_ios_telnet',
host='netacad11.info', port='####', timeout=60, username='admin',
password='cisco')
print(connection.send_command('show ip int brief'))
connection.disconnect()
```

Note: Sometimes it takes several tries to make the connection.

```
# Taken from Parts 4-8
# Connect to a Cisco Router using Telnet and run show commands
router = {
    'device_type': 'cisco_ios_telnet',
    'host': 'netacad11.info',
    'port': '####',
    'username': 'admin',
    'password': 'cisco',
    'secret': 'class',
}
```

Note: Port 23 is the default for Telnet, but a Terminal Server will assign a unique port to each device.

Optional – Backup the Configuration Files Using Telnet

1. From the python folder in a terminal, open VS Code:

   Code .

2. Reuse the text file named **hosts.txt** in the Python folder:

   Python > hosts.txt

3. Use the IP addresses of all the devices you want to get the configuration:

   192.168.1.1

   192.168.1.2

4. Save and close the file.
5. Create a new Python script named **backup-config-telnet.py**

   Python > backup-config-telnet.py

6. Add the following code:

```
#!/usr/bin/env python3

import getpass
import telnetlib

# Ask for username and password
user = input('Enter your telnet username: ')
password = getpass.getpass()

# Open a file called devices.txt
devices = open('hosts.txt')

#Telnet to devices
for line in devices:
    host = line.strip()
    tn = telnetlib.Telnet(host)

    tn.read_until(b'Username: ')
    tn.write(user.encode('ascii') + b'\n')
    if password:
        tn.read_until(b'Password: ')
        tn.write(password.encode('ascii') + b'\n')

    # Get the running configuration and save it to a file
    print('Getting the running config from IP: ' + (line))
    tn.write(b'terminal length 0\n')
    tn.write(b'show run\n')
    tn.write(b'exit\n')

    readoutput = tn.read_all()
    output = str(readoutput)
    saveoutput = open('Device_' + host + '.txt', 'wb')
    saveoutput.write(readoutput)
    saveoutput.close
```

7. Save and run the file:

   Run > Run Without Debugging