PYTHON FOR NETWORK AUTOMATION

INTERMEDIATE PYTHON

AUTHOR

Khaled Shemy

Index

Introduction

In today's fast-paced digital landscape, network automation has become a cornerstone for efficient network management and operation. Python, with its simplicity and versatility, has emerged as a preferred language for automating network tasks. This abstract explores real-life applications of Python in network automation, showcasing its transformative impact across various industries.

Python's extensive library ecosystem offers powerful tools for automating network configuration, monitoring, and troubleshooting. By leveraging libraries like Netmiko, jinja2, xlrd and schedule Python scripts can establish SSH connections to network devices, execute commands, and retrieve information, thereby streamlining repetitive tasks and reducing human error.

1-How to Generate configuration file and import it to your router??

Below code will generate Template configuration file by jinja2 library then push the template to the router using netmiko.

```
from netmiko import ConnectHandler
from jinja2 import Template

# Define the device details
device = {
    'device_type': 'cisco_ios',
    'host': '10.10.10.10',  # Replace with your device's IP
    'username': 'cisco',
    'password': 'cisco',

}
# Jinja2 configuration template
config_template = '''
hostname {{ device_hostname }}

ip vrf {{ vrf_name }}
 rd {{ rd_value }}
 route-target export {{ route_target_export }}
 route-target import {{ route_target_import }}
exit

router ospf {{ ospf_process_id }}
 router-id {{ ospf_router_id }}
 network {{ ospf_network }} {{ ospf_wildcard_mask }} area {{ ospf_area_id }}
exit

interface Loopback1
 ip address {{ loopback_ip }} {{ loopback_subnet_mask }}
exit

interface GigabitEthernet1/0
 ip address {{ interface_ip }} {{ interface_subnet_mask }}
 mpls ip
 no shutdown
exit

router bgp {{ bgp_autonomous_system }}
 neighbor {{ bgp_neighbor_ip }} remote-as {{ bgp_remote_as }}
 neighbor {{ bgp_neighbor_ip }}  update-source loopback1
 address-family vpnv4
  neighbor {{ bgp_neighbor_ip }} activate
 exit-address-family
 address-family ipv4 vrf {{ vrf_name }}
  redistribute connected
  redistribute static
 exit-address-family
exit
'''

# Define parameters
config_parameters = {
    'device_hostname': 'MyRouter',
    'vrf_name': 'my_vrf',
    'rd_value': '100:1',
    'route_target_export': '100:1',
    'route_target_import': '100:1',
    'ospf_process_id': 1,
    'ospf_router_id': '1.1.1.1',
    'ospf_network': '20.20.20.1',
    'ospf_wildcard_mask': '0.0.0.0',
    'ospf_area_id': 0,
    'bgp_autonomous_system': 65000,
    'bgp_neighbor_ip': '2.2.2.2',
    'bgp_remote_as': 65000,
    'interface_ip': '20.20.20.1',
    'interface_subnet_mask': '255.255.255.0',
    'loopback_ip': '1.1.1.1',
    'loopback_subnet_mask': '255.255.255.255',
}

# Create Jinja2 template and render the configuration
template = Template(config_template)
final_config = template.render(config_parameters)
print(final_config)
# Connect to the device
with ConnectHandler(**device) as net_connect:
    net_connect.enable()  # Enter privileged exec mode if required
    output = net_connect.send_config_set(final_config.splitlines())
```

2-How to Generate configuration file and import it to your router?? Use Chevron library

Below code will generate Template configuration file by chevron library

```
import chevron
# Define your configuration template using Chevron syntax
template = """
hostname {{ hostname }}
ip vrf {{ vrf_name }}
 rd  {{ vrf_id }}
 route-target export {{ rt_export }}
 route-target import {{ rt_import }}
exit
interface GigabitEthernet1/0
 description {{ interface_description }}
 ip vrf forwarding {{ vrf_name }}
 ip address {{ ip_address }} {{ subnet_mask }}
exit
router bgp {{ as_number }}
 bgp router-id {{ bgp_router_id }}
 neighbor {{ neighbor_ip }} remote-as {{ neighbor_as }}
 neighbor {{ neighbor_ip }} description {{ neighbor_description }}
 address-family ipv4 vrf {{ vrf_name }}
  redistribute static
  network {{ bgp_network }}
  neighbor {{ neighbor_ip }} remote-as {{ neighbor_as }}
  neighbor {{ neighbor_ip }} activate
  neighbor {{ neighbor_ip }} route-map {{ route_map }} in
 exit-address-family
exit
ip route vrf {{ vrf_name }} {{ destination_network }} {{ destination_subnet_mask }} {{ next_hop
}}
"""

# Define the data to be used in the template
data = {
    "hostname": "MyRouter",
    "vrf_name": "my_vrf",
    "vrf_id": "100:1",
    "rt_export": "100:1",
    "rt_import": "100:1",
    "interface_description": "Connection to Server",
    "ip_address": "192.168.2.1",
    "subnet_mask": "255.255.255.0",
    "as_number": "65000",
    "bgp_router_id": "1.1.1.1",
    "neighbor_ip": "192.168.2.2",
    "neighbor_as": "65001",
    "neighbor_description": "BGP Neighbor",
    "bgp_network": "10.0.0.0",
    "route_map": "MY_ROUTE_MAP",
    "destination_network": "192.168.3.0",
    "destination_subnet_mask": "255.255.255.0",
    "next_hop": "192.168.2.3",
}

# Render the template using Chevron
rendered_config = chevron.render(template, data)
print(rendered_config)
```

3-How to Generate configuration file and import it to your router?? Use Mako library

Below code will generate Template configuration file by mako library

```
from mako.template import Template
# Define your configuration template using Mako syntax
template = """
hostname ${hostname}
!
interface GigabitEthernet1/0
 description ${interface_description}
 ip address ${ip_address} ${subnet_mask}
exit
ip route ${destination_network} ${destination_subnet_mask} ${next_hop}
router ospf ${process}
 network ${ip_address} 0.0.0.0 area 0
exit
ip vrf ${vrf_name}
 rd ${rd_no}
exit
router bgp ${AS}
 neighbor 10.10.10.20 remote-as ${AS}
exit
"""

# Define the data to be used in the template
data = {
    "hostname": "MyRouter",
    "interface_description": "Connection to Server",
    "ip_address": "192.168.2.1",
    "subnet_mask": "255.255.255.0",
    "destination_network": "192.168.3.0",
    "destination_subnet_mask": "255.255.255.0",
    "next_hop": "192.168.2.3",
    "process": "1",
    "AS": 65000,
    "vrf_name": "service",
    "rd_no": "100:10"
}

# Create a Mako template object
mako_template = Template(template)

# Render the template with the data
rendered_config = mako_template.render(**data)
print(rendered_config)
```

4-How to simulate Flapping?

Sometimes you need to troubleshoot flapping, apply flapping to simulate the case

```
from netmiko import ConnectHandler
import time

# Replace these values with your router details
device = {
    'device_type': 'cisco_ios',
    'ip': '10.10.10.10',
    'username': 'cisco',
    'password': 'cisco',

}
ssh  = ConnectHandler(**device)
for _ in range(10):
    time.sleep(5)
    x =["interface Gigabitethernet1/0" , "shutdown","exit"]
    y = ssh.send_config_set(x)
    print(y)
    time.sleep(5)
    m = ["interface Gigabitethernet1/0", "no shutdown", "exit"]
    n = ssh.send_config_set(m)
    print(n)
```

5-How to take action during flapping?

If you are facing flapping on link and want to apply cost to shift traffic, simulate the case with below code

```
from netmiko import ConnectHandler
import time

# Replace these values with your router details
device = {
    'device_type': 'cisco_ios',
    'ip': '10.10.10.10',
    'username': 'cisco',
    'password': 'cisco',
}

ssh = ConnectHandler(**device)
x = 0

for _ in range(5):
    time.sleep(5)
    config_commands_shutdown = ["interface GigabitEthernet1/0", "shutdown", "exit"]
    result_shutdown = ssh.send_config_set(config_commands_shutdown)
    print(result_shutdown)

    time.sleep(5)
    config_commands_no_shutdown = ["interface GigabitEthernet1/0", "no shutdown", "exit"]
    result_no_shutdown = ssh.send_config_set(config_commands_no_shutdown)
    print(result_no_shutdown)

    x += 1

    # Break the loop and change IP when x reaches 2
    if x == 2:
        break

# Change IP after breaking out of the loop
time.sleep(5)
config_commands_change_ip = ["interface GigabitEthernet1/0", "ip address 60.60.60.1
255.255.255.0", "exit"]
result_change_ip = ssh.send_config_set(config_commands_change_ip)
print(result_change_ip)
```

6-How to automate commands by XML?

If your Team leader give you xml file and asked you to extract the output of the commands inside xml

```
import xml.etree.ElementTree as ET
from netmiko import ConnectHandler

# Parse the XML data
xml_data = """
<NetworkDevices>
    <NetworkDevice>
        <DeviceName>10.10.10.10</DeviceName>
        <Credentials>
            <Username>cisco1</Username>
            <Password>password1</Password>
        </Credentials>
        <DeviceType>cisco_ios</DeviceType>
        <Commands>
            <Command>show ip route</Command>
        </Commands>
    </NetworkDevice>
    <NetworkDevice>
        <DeviceName>10.10.10.20</DeviceName>
        <Credentials>
            <Username>cisco2</Username>
            <Password>password2</Password>
        </Credentials>
        <DeviceType>cisco_ios</DeviceType>
        <Commands>
            <Command>show ip int brief</Command>
            <Command>show ip int gig 0/0</Command>
        </Commands>
    </NetworkDevice>
    <NetworkDevice>
        <DeviceName>10.10.10.30</DeviceName>
        <Credentials>
            <Username>cisco3</Username>
            <Password>password3</Password>
        </Credentials>
        <DeviceType>cisco_ios</DeviceType>
        <Commands>
            <Command>show version</Command>
        </Commands>
    </NetworkDevice>
</NetworkDevices>

"""

root = ET.fromstring(xml_data)

# Loop through each NetworkDevice element and interact with routers
for network_device in root.findall('NetworkDevice'):
    device_name = network_device.find('DeviceName').text
    username = network_device.find('Credentials/Username').text
    password = network_device.find('Credentials/Password').text
    device_type = network_device.find('DeviceType').text
    commands = [command.text for command in network_device.findall('Commands/Command')]
    device = {
        'device_type': device_type,
        'ip': device_name,
        'username': username,
        'password': password,
    }

    with ConnectHandler(**device) as net_connect:
        print(f"Interacting with {device_name}...")
        for command in commands:
            output = net_connect.send_command(command)
            print(f"Command: {command}\n")
         print(f"Response:\n{output}\n")
```

7-XML with Netmiko (read from file)

You have file path for xml file and you need to automate the commands inside xml file

```
import xml.etree.ElementTree as ET
from netmiko import ConnectHandler
# Parse the XML data
fileee = open (r"C:\Users\khali\Music\sss.xml","r")
x = fileee.read()
root = ET.fromstring(x)
# Loop through each NetworkDevice element and interact with routers
for network_device in root.findall('NetworkDevice'):
    device_name = network_device.find('DeviceName').text
    username = network_device.find('Credentials/Username').text
    password = network_device.find('Credentials/Password').text
    device_type = network_device.find('DeviceType').text
    commands = [command.text for command in network_device.findall('Commands/Command')]
    device = {
        'device_type': device_type,
        'ip': device_name,
        'username': username,
        'password': password,
    }
    with ConnectHandler(**device) as net_connect:
        print(f"Interacting with {device_name}...")
        for command in commands:
            output = net_connect.send_command(command)
            with open(f"C:\\Users\\khali\\Music\\Sheet\\{device_name}_{command.replace('/', '-
')}.text", "w") as filee:
                filee.write("#" * 50 + "\n")
                filee.write(f"Command: {command}\n")
                filee.write(f"Response:\n{output}\n")
                filee.write("#" * 50 + "\n")
```

8-Create VLAN (1):

Create VLAN on three switches

```
from netmiko import ConnectHandler
devices = {
    "sw1" : "10.10.10.10",
    "sw2" : "10.10.10.20",
    "sw3" :  "10.10.10.30"
}

commonpara = {
      "device_type" : "cisco_ios",
      "username"  :  "admin",
      "password" :"admin"
}
vlan_commands = ["vlan 100" ,"name test","vlan 200","name shemy"]
for device_name, ip_address in devices.items():
    device_para =commonpara.copy()
    device_para.update({"ip":ip_address})
    print(f"connecting to {device_name} at { ip_address}")
    ssh = ConnectHandler(**device_para)
    output = ssh.send_config_set(vlan_commands)
    print(output)
    ssh.disconnect()
    print(f"Disconnected from {device_name}")
print("VLAN configuration completed on all switches")
```

9-Create VLAN (2):

Create VLAN using for method, apply below code

```
from netmiko import ConnectHandler

# Define device information
device = {
    'device_type': 'cisco_ios',
    'ip': '10.10.10.10',
    'username': 'cisco',
    'password': 'cisco',

}

# Define VLANs and descriptions
vlans = {
    2: 'VLAN 2 Description',
    3: 'VLAN 3 Description',
    4: 'VLAN 4 Description',
    5: 'VLAN 5 Description',
    6: 'VLAN 6 Description',
    7: 'VLAN 7 Description',
    8: 'VLAN 8 Description',
    9: 'VLAN 9 Description',
    10: 'VLAN 10 Description',
    11: 'VLAN 11 Description',
    12: 'VLAN 12 Description',
}

# Connect to the device
with ConnectHandler(**device) as net_connect:
    net_connect.enable()

    # Loop through VLANs and apply configurations
    for vlan_id, description in vlans.items():
        config_commands = [
            f'vlan {vlan_id}',
            f'name {description}',
            'exit',
        ]
        output = net_connect.send_config_set(config_commands)
        print(output)
        print(f"Configured VLAN {vlan_id} with description: {description}")
```

10-Extend VLAN and check if repeated

Imagine you have ring topology (including switches) and each switch connected with group of customers (VLANS), You want to extend New VLAN for New customer, the code working as below:

-If you want to delete customer it will delete specified VLAN.

-If you want to extend new VLAN make sure VLAN not repeated in the ring, if not repeated extend the VLAN, if not repeated don't extend the VLAN

```
from netmiko import ConnectHandler
from netmiko.ssh_exception import NetMikoTimeoutException, NetMikoAuthenticationException

try:
    Action = input("Input Action: ")
    vlan = input("Input new VLAN: ")
    IP = input("Input IP: ")
    Interface = input("Input Interface: ")

    Shemy = ["10.10.10.10", "10.10.10.20", "10.10.10.30"]

    if Action == "remove":
        device = {
            "device_type": "cisco_ios",
            "ip": IP,
            "username": "cisco",
            "password": "cisco"
        }
        ssh = ConnectHandler(**device)
        Config = [
            f"interface {Interface}",
            f"switchport trunk allowed vlan {Action} {vlan}",
            "exit",
            f"no vlan {vlan}"
        ]
        ssh.send_config_set(Config)

    elif Action == "add":
        vlan_found = False

        for switch_ip in Shemy:
            device = {
                "device_type": "cisco_ios",
                "ip": switch_ip,
                "username": "cisco",
                "password": "cisco"
            }
            ssh = ConnectHandler(**device)
            prompt = ssh.find_prompt()
            newprompt = prompt.strip("#")
            show_vlan_command = f"show vlan id {vlan}"
            show_vlan_output = ssh.send_command(show_vlan_command)

            if f"VLAN id {vlan} not found" not in show_vlan_output:
                vlan_found = True
                break

        if not vlan_found:
            device_add = {
                "device_type": "cisco_ios",
                "ip": IP,
                "username": "cisco",
                "password": "cisco"
            }
            ssh_add = ConnectHandler(**device_add)
            Config = [
                f"vlan {vlan}",
                "exit",
                f"interface {Interface}",
                f"switchport trunk allowed vlan add {vlan}",
                "exit"
            ]
            ssh_add.send_config_set(Config)
            print(f"VLAN {vlan} added successfully to {IP}")
        else:
            print(f"No need to take action. VLAN {vlan} exists on {newprompt}.")

except (NetMikoTimeoutException, NetMikoAuthenticationException) as e:
    print(f"Error occurred: {e}. Connection failed.")
except Exception as e:
    print(f"Error occurred: {e}")
```

11-Extend VlAN using While method(1):

To Automate VLAN using while method, use below Code:

```
from netmiko import ConnectHandler
switch_num = int(input("How many switches would like to configure:"))
User= input("Username:")
Password= input("Enter Passwword:")
while switch_num > 0 :
    hostip = input("Switch ip:")
    Switch ={
        "device_type":"cisco_ios",
        "ip" : hostip,
        "username" : User,
        "password" : Password
    }
    ssh =ConnectHandler(**Switch)
    hostname = ssh.send_command("show run | inc host")
    x = hostname.split()
    print(x)
    device = x[1]
    print(device)
    int_range =input ("Enter Interface Range:")
    trunk= [f"interface range {int_range}","switchport  trunk encapsulation dot1q ","switchport
mode trunk","switchport trunk allowed vlan all","no shutdown"]
    output = ssh.send_config_set(trunk)
    print(output)
    switch_num -= 1
    print(f"configured {device}  For trunk interface range")

input("Press Enter to Continue")
```

12-Extend VlAN using While method(2):

```
from netmiko import ConnectHandler
switch_num = int(input("How many switches would like to configure:"))
User= input("Username:")
Password= input("Enter Passwword:")
while switch_num > 0 :
    hostip = input("Switch ip:")
    Switch ={
        "device_type":"cisco_ios",
        "ip" : hostip,
        "username" : User,
        "password" : Password
    }
    ssh =ConnectHandler(**Switch)
    hostname = ssh.send_command("show run | inc host")
    x = hostname.split()
    print(x)
    device = x[1]
    vlan_num = input(" How many VLAN want to extend ?")
    vlan_num =  int(vlan_num)
    while vlan_num > 0 :
      vlan = input("VLAN ID:")
      int_rang = input("Enter Range of interfaces")
      int_rang_command = "interface range " +  int_rang
      access_command = "switchport access vlan " + vlan
      config_commands = [int_rang_command,"switchport mode access",access_command]
      y = ssh.send_config_set(config_commands)
      print(y)
      vlan_num -=1
      print(f"switch {device} configured")
      print("#"*70)
    switch_num -=1
input("Press Enter to finish")
```

13-Check BGP status report for all routers in the network

Below report will make report in csv about BGP status for the routers

Note: For template file

textfsm3.template:
1-Apply text file and write the below:

Value NEIG (\S+)
Value State (\S+)

Start
^${NEIG}\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\d+\s+\S+\s+${State} -> Record

2- go cmd then type: cd C:\Users\khali\OneDrive\Desktop\ISP Traffic then press enter

3- apply on cmd : rename textfsm3.txt to textfsm3.template

```
import csv
import textfsm
from netmiko import ConnectHandler

devices = [
    {
        "device_type": "cisco_ios",
        "ip": "10.10.10.10",
        "username": "cisco",
        "password": "cisco"
    },
    {
        "device_type": "cisco_ios",
        "ip": "10.10.10.20",
        "username": "cisco",
        "password": "cisco"
    },
    {
        "device_type": "cisco_ios",
        "ip": "10.10.10.30",
        "username": "cisco",
        "password": "cisco"
    }
]

# Store parsed data for all routers
all_parsed_data = []

for device in devices:
    ssh = ConnectHandler(**device)
    with open(r'C:\Users\khali\OneDrive\Desktop\ISP Traffic\textfsm3.template', 'r') as
template_file:
        template = textfsm.TextFSM(template_file)

    router_output = ssh.send_command("show ip bgp summary")
    parsed_data = template.ParseText(router_output)

    if parsed_data:
        modified_data = [[device['ip'], row[0], 'Established' if any(char.isdigit() for char in
row[1]) else row[1]] for row in parsed_data]
        all_parsed_data.extend(modified_data)

# Write data to a CSV file
with open(r'C:\Users\khali\OneDrive\Desktop\Session\Shemy2.csv', 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)
    csv_writer.writerow(["Router IP", "Remote End", "State"])
    csv_writer.writerows(all_parsed_data)

print("Data written to Shemy2.csv")
```

14-Change show ip ospf nei and add area on the table

Below report will make report in csv about OSPF status for the routers plus adding area

```
import re
from netmiko import ConnectHandler
from tabulate import tabulate
import pandas as pd
import socket

def resolve_hostname(ip_address):
    try:
        hostname = socket.gethostbyaddr(ip_address)[0]
        return hostname
    except socket.herror:
        return ip_address

devices = [
    {
        'device_type': 'cisco_ios',
        'ip': '10.10.10.10',
        'username': 'cisco',
        'password': 'cisco',
    },
    {
        'device_type': 'cisco_ios',
        'ip': '10.10.10.20',
        'username': 'cisco',
        'password': 'cisco',
    },
    {
        'device_type': 'cisco_ios',
        'ip': '10.10.10.30',
        'username': 'cisco',
        'password': 'cisco',
    }
]

data = []

for device in devices:
    try:
        net_connect = ConnectHandler(**device)
        shemy = net_connect.find_prompt()
        output = net_connect.send_command("show ip ospf nei")

        ospf_nei_pattern =
re.compile(r"(\d+\.\d+\.\d+\.\d+)\s+(\d+)\s+(\S+)\s+(\S+)\s+(\d+\.\d+\.\d+\.\d+)\s+(\S+)")
        ospf_nei_matches = ospf_nei_pattern.findall(output)

        for entry in ospf_nei_matches:
            router_ip = device['ip']
            neighbor_ip = entry[0]
            interface = entry[5]
            show_ospf_interface_cmd = f"show ip ospf interface {interface} | include Area"
            ospf_interface_output = net_connect.send_command(show_ospf_interface_cmd)

            area_pattern = re.compile(r'Internet
Address\s+\d+.\d+\d+.\d+.\d+\D\d+\W\s+\S+\s+(.+)\W\s\S+\s+\S+\s+\S+\s+\S+')
            area_match = area_pattern.search(ospf_interface_output)
            area = area_match.group(1) if area_match else 'N/A'

            data.append([router_ip, neighbor_ip] + list(entry[1:5]) + [interface, area])

    except Exception as e:
        print(f"Error connecting to the device {device['ip']}: {e}")

    finally:
        net_connect.disconnect()

df = pd.DataFrame(data, columns=["Router IP", "Neighbor IP", "Pri", "State", "Dead Time", "Address",
"Interface", "Area"])

table = tabulate(df, headers='keys', tablefmt='plain', showindex=False)

with open(r"C:\Users\khali\OneDrive\Desktop\ISP Traffic\show_ospf.csv", "w") as file:
    file.write(table)

print(table)
```

15-Automte commands written by excel sheets

Automate commands written inside excel sheet

```
import xlrd
from netmiko import ConnectHandler

wrk123 = xlrd.open_workbook_xls(r"C:\Users\khali\Music\NetworkingGiants.xls")
sheet123 = wrk123.sheet_by_name("Khaled")

for index123 in range(1, sheet123.nrows):
    ip = sheet123.row(index123)[1].value
    Hostname = sheet123.row(index123)[0].value
    device_type = sheet123.row(index123)[4].value
    Username = sheet123.row(index123)[2].value
    Password = sheet123.row(index123)[3].value
    Commands = sheet123.row(index123)[5].value
    Commandss = Commands.splitlines()

    for Commandsss in Commandss:
        devices = {
            "device_type": device_type,
            "ip": ip,
            "username": Username,
            "password": Password,
        }
        ssh123 = ConnectHandler(**devices)
        show = ssh123.send_config_set(Commandsss)

        # Replace "gig0/0" with "gig0-0" in the file path
        file_path =
fr"C:\\Users\\khali\\Music\\Sheet\\sheet_{devices['device_type']}{devices['ip']}{Hostname}_{Comm
andsss.replace('/', '-')}.txt"
        file = open(file_path, "w")

        Test1 = file.write("#" * 50 + "\n")
        Test2 = file.write(f" Results for {Hostname} :" + "\n")
        Test3 = file.write(f"Commands {Commandsss} :" + "\n")
        Test4 = file.write(f"{show}" + "\n")
        Test6 = file.write("#" * 50 + "\n")
```

16-Automate commands written by YAML

You have file path for Yaml file and you need to automate the commands inside Yaml file

```
import yaml
from netmiko import ConnectHandler


# Function to execute commands on a device
def execute_commands(device_info):
    try:
        # Remove 'commands' from device_info temporarily
        commands = device_info.pop('commands')

        # Establish SSH connection to the device
        connection = ConnectHandler(**device_info)

        # Get the IP address of the device
        ip_address = device_info['ip']


        # Create a file for each IP address
        prompt = connection.find_prompt().strip("#")
        filename = f'C:\\Users\\khali\\OneDrive\\Desktop\\NGN-{ip_address}-{prompt}.txt'
        with open(filename, 'w') as file:
            for command in commands:
                output = connection.send_command(command)

                print(prompt)

                # Write the command and its output to the file
                file.write("#" * 10 + "\n")
                file.write(f"Command: {command}\n")
                file.write(output + "\n")
                file.write("#" * 10 + "\n")

        # Close the SSH connection
        connection.disconnect()
    except Exception as e:
        print(f"Error connecting to {device_info['ip']}: {str(e)}")


if __name__ == "__main__":
    # Load device information from the YAML file
    with open(r'C:\Users\khali\OneDrive\Desktop\khaled123.yml', 'r') as file:
        devices = yaml.safe_load(file)

    # Iterate through devices and execute commands
    for device in devices:
        execute_commands(device)
```

17-Automate commands written by Json (read from file)

You have file path for Json file and you need to automate the commands inside Json file

```
from netmiko import ConnectHandler
import json

# Load router configurations from a JSON file
with open(r"C:\Users\khali\OneDrive\Desktop\khaled123.json", "r") as x:
    router_configs = json.load(x)

# Iterate through router configurations
for router_config in router_configs:
    try:
        # Connect to the router using SSH
        connection = ConnectHandler(
            device_type=router_config['device_type'],
            ip=router_config['ip'],
            username=router_config['username'],
            password=router_config['password']
        )

        # Execute commands and capture output
        results = {}
        for command in router_config['commands']:
            output = connection.send_command(command)
            results[command] = output

        # Save results to a file with IP address as the filename
        filename = fr"C:\Users\khali\Music\MSAN\{router_config['ip']}_results.txt"
        with open(filename, 'w') as result_file:
            result_file.write(f"Results for {router_config['device_type']} at
{router_config['ip']}:\n")
            for command, output in results.items():
                result_file.write("#" * 10 + "\n")
                result_file.write(f"Command: {command}\nOutput:\n{output}\n")
                result_file.write("#" * 10 + "\n")

        # Disconnect from the router
        connection.disconnect()

    except Exception as e:
        print(f"An error occurred for {router_config['device_type']} at {router_config['ip']}:
{str(e)}"
```

18-Each user will login and generate command on the router

imagine you have multiple account and each account responsible for multiple commands to automate

```
from netmiko import ConnectHandler

# List of devices with their corresponding usernames and passwords
devices = [
    {'username': 'cisco1', 'password': 'cisco1'},
    {'username': 'cisco2', 'password': 'cisco2'},
    {'username': 'cisco3', 'password': 'cisco3'},
    {'username': 'cisco4', 'password': 'cisco4'},
]

# List to store Netmiko connection objects
connections = []

# Loop through devices and establish SSH sessions
for device_info in devices:
    device = {
        'device_type': 'cisco_ios',
        'ip': '10.10.10.10',
        'username': device_info['username'],
        'password': device_info['password'],
    }
    connection = ConnectHandler(**device)

    # Create a filename with a raw string
    filename = fr"C:\Users\khali\OneDrive\Desktop\ISP
Traffic\Files\{device_info['username']}.txt"

    with open(filename, "w") as file:
        if device_info['username'] == "cisco1":
            x = connection.send_command("show ip interface brief")
            file.write(x)
            print(x)
        elif device_info['username'] == "cisco2":
            y = connection.send_command("show ip route")
            file.write(y)
            print(y)
        elif device_info['username'] == "cisco3":
            z = connection.send_command("show ip vrf")
            file.write(z)
            print(z)
        elif device_info['username'] == "cisco4":
            d = connection.send_command("show version")
            file.write(d)
            print(d)

# Close all SSH sessions
for connection in connections:
    connection.disconnect()
```

19-Taking Backup using Telnet-1

Below Two codes to collect backup using Telnet

```
from netmiko import ConnectHandler
# Define device information
switch = ["10.10.10.10","10.10.10.20","10.10.10.30","10.10.10.40"]
for x in switch:
 if x== "10.10.10.10" :
    device = {
        "device_type": "cisco_ios",
        "ip": "10.10.10.10",
        "username": "cisco",
        "password": "cisco"
    }
    ssh1 = ConnectHandler(**device)
    a = ssh1.send_command("show running-config")
    file1 = open(f"C:\\Users\\khali\\OneDrive\\Desktop\\ISP Traffic\\{device["ip"]}.txt","w")
    aa = file1.write(a)
 elif x== "10.10.10.20" :
    device = {
        "device_type": "cisco_ios",
        "ip": "10.10.10.20",
        "username": "cisco",
        "password": "cisco"
    }
    ssh2 = ConnectHandler(**device)
    b = ssh2.send_command("show running-config")
    file1 = open(f"C:\\Users\\khali\\OneDrive\\Desktop\\ISP Traffic\\{device["ip"]}.txt", "w")
    bb = file1.write(b)
elif x== "10.10.10.30" :
    device = {
        "device_type": "cisco_ios_telnet",
        "ip": "10.10.10.30",
        "username": "cisco",
        "password": "cisco"
    }
    telnet1 = ConnectHandler(**device)
    c = telnet1.send_command("show running-config")
    file1 = open(f"C:\\Users\\khali\\OneDrive\\Desktop\\ISP Traffic\\{device["ip"]}.txt", "w")
    cc= file1.write(c)
elif x== "10.10.10.40" :
    device = {
        "device_type": "cisco_ios_telnet",
        "ip": "10.10.10.40",
        "username": "cisco",
        "password": "cisco"
    }
    telnet2 = ConnectHandler(**device)
    d = telnet2.send_command("show running-config")
    file1 = open(f"C:\\Users\\khali\\OneDrive\\Desktop\\ISP Traffic\\{device["ip"]}.txt", "w")
    dd = file1.write(d)
```

20-Taking Backup using Telnet-2

```
from netmiko import ConnectHandler

switch1 = ["10.10.10.10","10.10.10.20"]# only support ssh
switch2 = ["10.10.10.30","10.10.10.40"]# only support telnet
x = "cisco_ios"
y = "cisco_ios_telnet"

if x =="cisco_ios" :
     for aa in switch1 :
        device = {
            "device_type": x,
            "ip": aa,
            "username": "cisco",
            "password": "cisco"
        }
        cc = ConnectHandler(**device)
        yy =cc.send_command("show running-config")
        print(yy)
if y == "cisco_ios_telnet":
        for bb in switch2 :
         device = {
           "device_type": y,
           "ip": bb,
           "username": "cisco",
           "password": "cisco",

         }
         dd = ConnectHandler(**device)
         ee = dd.send_command("show running-config")
         print(ee)
```

21-Automate Backup for multiple device types:

Imagine you have multi vendor with different username and password and you have to manage backup and save the configuration for each vendor in different file.

```
from netmiko import ConnectHandler
from netmiko.ssh_exception import NetMikoTimeoutException, NetMikoAuthenticationException

# Create a list of the network devices that you want to back up.
devices = [
    {
        "device_type": "cisco_ios",
        "ip": "10.10.10.10",
        "username": "cisco1",
        "password": "password1"
    },
    {
        "device_type": "fortinet",
        "ip": "10.10.10.60",
        "username": "admin",
        "password": ""
    },
    {
        "device_type": "nokia_sros",
        "ip": "10.10.10.1",
        "username": "admin",
        "password": "admin"
    }
]

# Iterate over the list of devices and connect to each device using Netmiko.
for device in devices:
    try:
        net_connect = ConnectHandler(**device)

        # Use Netmiko to execute the backup command for the device.
        if device["device_type"] == "cisco_ios":
            backup_command = "show running-config"
            shemy = net_connect.find_prompt().strip("#")
            backup_output = net_connect.send_command(backup_command)
            with
open(f"C:\\Users\\khali\\Music\\Cisco\\backup_{device['device_type']}_{device['ip']}_{shemy}.txt", "w")
as Test:
                Test.write(backup_output)
        elif device["device_type"] == "fortinet":
            backup_command = "show full-configuration"
            shemy = net_connect.find_prompt().strip("#")
            backup_output = net_connect.send_command(backup_command)
            with
open(f"C:\\Users\\khali\\Music\\Fortinet\\backup_{device['device_type']}_{device['ip']}_{shemy}.txt",
"w") as Test:
                Test.write(backup_output)
        elif device["device_type"] == "nokia_sros":
            backup_command = "admin display-config"
            shemy = net_connect.find_prompt().strip("*:A#")
            backup_output = net_connect.send_command(backup_command)
            with
open(f"C:\\Users\\khali\\Music\\Nokia\\backup_{device['device_type']}_{device['ip']}_{shemy}.txt", "w")
as Test:
                Test.write(backup_output)

        # Disconnect from the device.
        net_connect.disconnect()
    except NetMikoTimeoutException as e:
        print(f"Timeout error for device {device['ip']}_{device['device_type']} : {e}" + "\n")
        shemy2 = open(r"C:\Users\khali\Music\Timeout-issue.txt", "a")
        copy2 = shemy2.write(f"Timeout error for device {device['ip']}_{device['device_type']} : {e}" +
"\n")
    except NetMikoAuthenticationException as e:
        print(f"Timeout error for device {device['ip']}_{device['device_type']} : {e}" + "\n")
        shemy3 = open(r"C:\Users\khali\Music\AuthenticationException.txt", "a")
        copy3 = shemy3.write(f"Timeout error for device {device['ip']}_{device['device_type']} : {e}" +
"\n")
    except Exception as e:
        print(f"An error occurred for device {device['ip']}_{device['device_type']} : {e}" + "\n")
        shemy4 = open(r"C:\Users\khali\Music\other issue.txt", "a")
        copy4 = shemy4.write(f"Timeout error for device {device['ip']}_{device['device_type']} : {e}" +
"\n")
```

22-Check Visibility

Below code will check the visibility for each router

```
from netmiko import ConnectHandler

# Define device information
devices = [
    {
        'device_type': 'cisco_ios',
        'ip': 'router1_ip',
        'username': 'username1',
        'password': 'password1',
        'secret': 'enable_password1',
    },
    {
        'device_type': 'cisco_ios',
        'ip': 'router2_ip',
        'username': 'username2',
        'password': 'password2',
        'secret': 'enable_password2',
    },
    {
        'device_type': 'cisco_ios',
        'ip': 'router3_ip',
        'username': 'username3',
        'password': 'password3',
        'secret': 'enable_password3',
    },
]

# Check visibility by pinging and write results to a text file
with open('visibility_results.txt', 'w') as file:
    for device in devices:
        try:
            with ConnectHandler(**device) as net_connect:
                ping_result = net_connect.send_command(f'ping {device["ip"]}')
                if 'Success rate is 0 percent' in ping_result:
                    file.write(f"{device['ip']} Router is invisible\n")
                else:
                    file.write(f"{device['ip']} Router is visible\n")
        except Exception as e:
            file.write(f"{device['ip']} Router is invisible\n")
```

23-Error Handling:

Below code will present if you are facing issue in access either Time out or Authentication

Note: you need to downgrade netmiko to 3.4.0 , to make sure the code working properly

```
from netmiko import ConnectHandler
from netmiko.ssh_exception import NetMikoTimeoutException,NetMikoAuthenticationException
w = open(r"C:\Users\khali\Music\shemy.txt", "r")
ww = w.read().split(",")
print(ww)
for singledevice in ww:
    devicedictionary = {
        "ip": singledevice,
        "username": input("enter the username for " + singledevice + ":"),
        "password": "cisco",
        "device_type": "cisco_ios",
        "secret": "cisco",
    }
    try:
        ssh123 = ConnectHandler(**devicedictionary) #*arg = set
        prompt123 = ssh123.find_prompt()
        newprompt123 = prompt123.strip("#")
        print(newprompt123)
        #ssh established
        output123 = ssh123.send_command("show int description")
        print(output123)
        shemy = open(r"C:\Users\khali\Music\Working.txt", "a")
        copy1 = shemy.write(singledevice + "\n")
    except NetMikoTimeoutException:
        print(singledevice + " this has Timeout Issue")
        shemy2 = open(r"C:\Users\khali\Music\Timeout-issue.txt", "a")
        copy2 = shemy2.write(singledevice + "\n")
    except NetMikoAuthenticationException:
        print(singledevice + " this has Authentication Issue")
        shemy3 = open(r"C:\Users\khali\Music\AuthenticationException.txt", "a")
        copy3 = shemy3.write(singledevice + "\n")
```

24-Automate Backup every one min.

If we need to check during traffic at night time (Python can give you solution for this to automate commands every day, every week, every one mins)

```
from netmiko import ConnectHandler
from datetime import datetime
import schedule
import time

ssh = open(r"C:\Users\khali\Music\split.txt", "r")
w = ssh.read().split(",")


def job():
    for single in w:
        device = {
            "ip": single,
            "username": "admin",
            "password": "cisco",
            "device_type": "cisco_ios",
            "secret": "cisco123"
        }
        ssh123 = ConnectHandler(**device)
        prompt123 = ssh123.find_prompt()
        newprompt123 = prompt123.strip("#")

        khaled = ssh123.send_command("show running-config")

        x = datetime.now()
        customize = str(x.year) + "-" + str(x.month) + "-" + str(x.day) + "_" + str(x.hour) +
"_" + str(x.minute)
        file = open(r"C:\Users\khali\Music\NGN-Project" + "_" + single + "_" + newprompt123 +
"_" + customize + ".txt",
                    "a")
        file.write("#" * 10 + "\n")
        file.write("show running-config" + "\n")
        file.write(khaled + "\n")
        file.write("#" * 10 + "\n")
        file.close()
ssh123.disconnect()


schedule.every(60).seconds.do(job)
while True:
    schedule.run_pending()
    time.sleep(1)
```

25-Pythcon code to check common configuration file between two routers:

To do Comparison and check the common configuration between two routers.

```
from netmiko import ConnectHandler
import difflib

# Device parameters
device1_params = {
    'device_type': 'cisco_ios',
    'ip': '10.10.10.10',
    'username': 'cisco',
    'password': 'cisco',
}

device2_params = {
    'device_type': 'cisco_ios',
    'ip': '10.10.10.20',
    'username': 'cisco',
    'password': 'cisco',
}

# Establish connections to devices
device1_connection = ConnectHandler(**device1_params)
device2_connection = ConnectHandler(**device2_params)

# Retrieve running configurations
running_config_device1 = device1_connection.send_command("show running-config")
running_config_device2 = device2_connection.send_command("show running-config")

# Close connections
device1_connection.disconnect()
device2_connection.disconnect()

# Save configurations to files
with open("device1_config.txt", "w") as file1:
    file1.write(running_config_device1)

with open("device2_config.txt", "w") as file2:
    file2.write(running_config_device2)

# Compare configurations
with open("device1_config.txt") as file1, open("device2_config.txt") as file2:
    config1 = file1.readlines()
    config2 = file2.readlines()

differ = difflib.Differ()
diff = list(differ.compare(config1, config2))

# Print only the common lines (lines that do not start with '-', '+', or '?')
common_lines = [line for line in diff if line.startswith("  ")]

with open(r'C:\Users\khali\OneDrive\Desktop\common.txt','w') as file:
 file.writelines(common_lines)
```

26-Pythcon code to check different configuration file between two routers:

To do comparison and check the difference configuration between two routers.

```
from netmiko import ConnectHandler
import difflib

# Device parameters
device1_params = {
    'device_type': 'cisco_ios',
    'ip': '10.10.10.10',
    'username': 'cisco',
    'password': 'cisco',
}

device2_params = {
    'device_type': 'cisco_ios',
    'ip': '10.10.10.20',
    'username': 'cisco',
    'password': 'cisco',
}

# Establish connections to devices
device1_connection = ConnectHandler(**device1_params)
device2_connection = ConnectHandler(**device2_params)

# Retrieve running configurations
running_config_device1 = device1_connection.send_command("show running-config")
running_config_device2 = device2_connection.send_command("show running-config")

# Close connections
device1_connection.disconnect()
device2_connection.disconnect()

# Save configurations to files
with open("device1_config.txt", "w") as file1:
    file1.write(running_config_device1)

with open("device2_config.txt", "w") as file2:
    file2.write(running_config_device2)

# Compare configurations
with open("device1_config.txt") as file1, open("device2_config.txt") as file2:
    config1 = file1.readlines()
    config2 = file2.readlines()

differ = difflib.Differ()
diff = list(differ.compare(config1, config2))

with open(r'C:\Users\khali\OneDrive\Desktop\trtr.txt','w') as file:
    file.writelines(diff)
```

27-Automate default route in case Primary link is down

Below code will Automate default route if Primary link is down

```
from netmiko import ConnectHandler
import time

device = {
    "device_type": "cisco_ios",
    "ip": "1.1.1.1",
    "username": "cisco",
    "password": "cisco"
}
ssh = ConnectHandler(**device)
ping_result = ssh.send_command("ping 30.30.30.2")

if "Success rate is 100 percent" in ping_result:
    ssh.send_config_set("ip route 0.0.0.0 0.0.0.0 30.30.30.2")
    print("Static route created for Primary Link")
    print("Waiting for 20 seconds before continuous checking of primary link...")
    time.sleep(20)  # Wait for 20 seconds before continuous checking
    while True:
        ping_result = ssh.send_command("ping 30.30.30.2")
        if "Success rate is 0 percent" in ping_result:
            ssh.send_config_set("ip route 0.0.0.0 0.0.0.0 40.40.40.2 5")
            print("Static route created for Secondary Link")
            break  # Exit the loop once secondary route is configured
        else:
            print("Primary link still up. Checking again in 20 seconds...")
            time.sleep(20)  # Wait for 20 seconds before checking again
else:
    print("Primary link down. Static route for secondary link not configured.")
```

28-Automate port-security configuration using while method

Below code will show how to automate port-security setup in case sticky or static

```
from netmiko import ConnectHandler
switch_num = input("How switch would like to configure?")
switch_num = int(switch_num)
USER = input("Enter username:")
Pass = input("Enter Password:")
while switch_num > 0 :
    hostip = input("Switch IP:")
    switch = {
        "device_type":"cisco_ios",
        "ip": hostip,
        "username": USER,
        "password": Pass
    }
    ssh =ConnectHandler(**switch)
    hostname = ssh.send_command("show run | inc host")
    x = hostname.split()
    device = x[1]
    sticky = input("would you like to configure sticky mode : [y/n]")
    if sticky.lower() == "y":
        interface = input("Enter the range of interfaces:")
        intreface_command = "interface range " + interface
        config = [intreface_command, "switchport mode access","switchport port-security" ,
"switchport port-security mac sticky","switchport port-security violation shutdown"]
        output = ssh.send_config_set(config)
        print(output)
        print(f"switch {device} configured")
        print("-" * 79)
    else:
        int_num = input("How many interfaces need to configure as static port-security mode?")
        int_num = int(int_num)
        while int_num > 0 :
            int_id = input("Enter the interface to be in static mode")
            int_id_command = "interface " + int_id
            MAC = input("Enter MAC address for " + int_id)
            port_security_command = "switchport port-security mac " +MAC
            Command = [int_id_command,"switchport mode access","switchport port-security" ,
port_security_command, "switchport port-security violation shutdown"]
            output = ssh.send_config_set(Command)
            print(output)
            print(f"switch {device} configured")
            print("-" * 79)
            int_num -=1
    switch_num -=1
input("Press Enter to finish")
```

29-OSPF Confirmation message

Below code will make confirmation message before applying OSPF configuration.

```
from netmiko import ConnectHandler

# Define device information
device = {
    'device_type': 'cisco_ios',
    'host': 'x.x.x.x',
    'username': 'cisco',
    'password': 'cisco',
}

ospf_config = []

while True:
    process_id = input("Enter OSPF process ID: ")
    network = input("Enter network address: ")
    wildcard_mask = input("Enter wildcard mask: ")
    area = input("Enter OSPF area: ")

    ospf_config.append((process_id, network, wildcard_mask, area))

    confirm = input("Are you sure about the configuration? (y/n): ")
    if confirm.lower() == 'y':
        print("\nOSPF Configuration:")
        for config in ospf_config:
            print(f"Process ID: {config[0]}, Network: {config[1]}, Wildcard Mask: {config[2]},
Area: {config[3]}")

        # Push OSPF configuration to the router
        with ConnectHandler(**device) as net_connect:
            config_commands = [f"router ospf {process_id}",
                               f"network {network} {wildcard_mask} area {area}"
                               ]
            output = net_connect.send_config_set(config_commands)
            print("\nConfiguration pushed to the router successfully.")
        break
    elif confirm.lower() == 'n':
        print("Need to review the configuration.")
        break
    else:
        print("Invalid input. Please enter 'y' or 'n'.")
```

30-collect Logs in case BGP Down

Below code will collect logs in case BGP is down

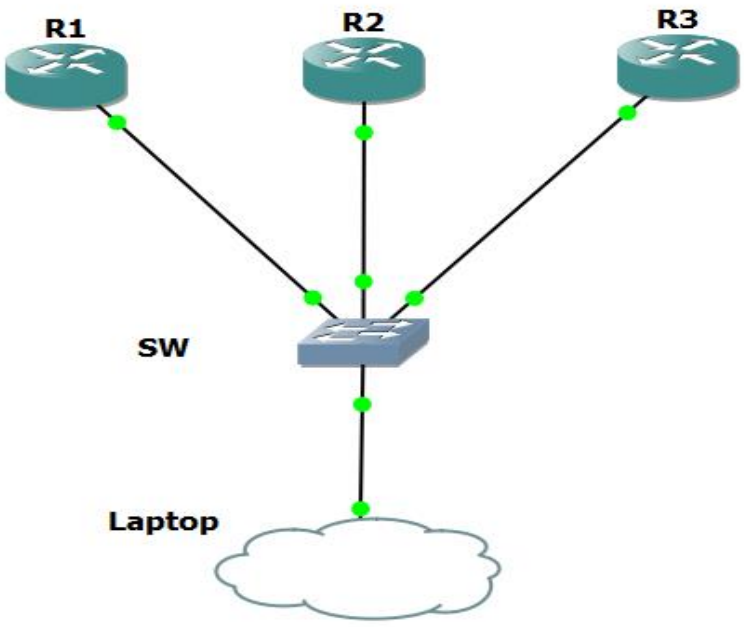
```
from netmiko import ConnectHandler
import time

device = {
          "device_type": "cisco_ios",
           "ip":"x.x.x.x",
           "username":"cisco",
           "password":"cisco",
}

ssh = ConnectHandler(**device)
command = "show ip bgp summary"
output = ssh.send_command(command)
time.sleep(20)
if "Connent" in output or "Idle" in output or "Active" in output:  # Fixing condition
    logs = ["show arp","show ip route","show ip int brief","show ip bgp neighbors","show
running-config | section bgp","show ip bgp"]
    for x in logs:
        y = ssh.send_command(x)
        print("#" * 60)
        print(x + ":")
        print(y)
```

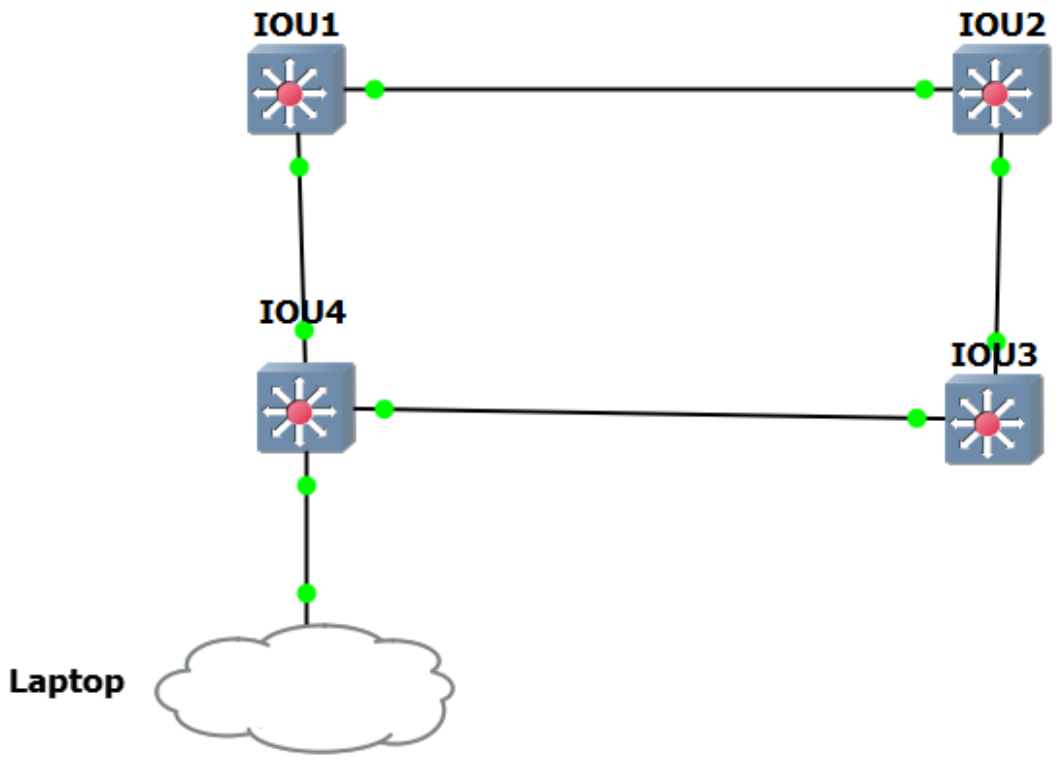
31- Collect logs when BGP is down (textfsm used)

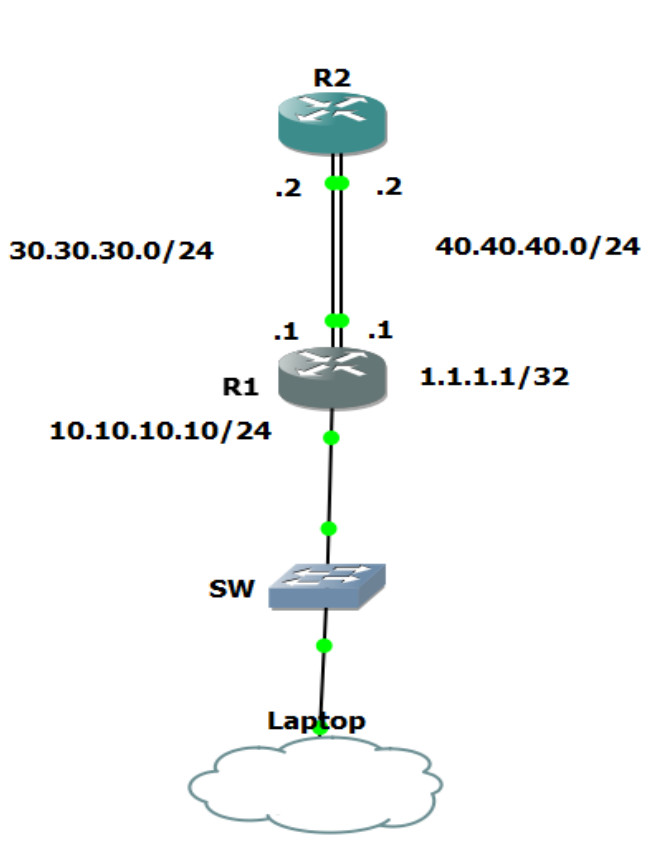
Below code will collect logs in case BGP is down (with textfsm)

```
from netmiko import ConnectHandler
device = {
    "device_type": "cisco_ios",
    "ip": "10.10.10.10",
    "username": "cisco",
    "password": "cisco",
}
ssh = ConnectHandler(**device)
command = "show ip bgp summary"
output = ssh.send_command(command, use_textfsm=True)
print(output)
# Check BGP state and apply logic
bgp_state = output[0]['state_pfxrcd']
if bgp_state in ['Connect', 'Active', 'Idle']:
    for entry in output:
        bgp_neighbor = entry['bgp_neigh']
        logs = ["show arp", f"show ip route {bgp_neighbor}", "show ip int brief",
                f"show ip bgp neighbors {bgp_neighbor}", "show running-config | section bgp",
                "show ip bgp"]
        for x in logs:
            y = ssh.send_command(x)
            print("#" * 60)
            print(x + ":")
            print(y)
else:
    print('BGP is up')
```

Topology


R1
R2
R3
SW
Laptop


IOU1
IOU2
IOU4
IOU3
Laptop


R2
.2
.2
30.30.30.0/24
40.40.40.0/24
.1
.1
R1
1.1.1.1/32
10.10.10.10/24
SW
Laptop

Excel Sheet(xlrd library)

Hostname	IP	Username	Password	device type	Commands
cisco	10.10.10.10	cisco	cisco	cisco_ios	do show ip route do show int des
nokia	10.10.10.30	cisco	cisco	cisco_ios	do show version do show ip interface brief
huawei	10.10.10.20	cisco	cisco	cisco_ios	do show vlan brief do show ip interface brief

Yaml(.yml)

```
- device_type: cisco_ios
  ip: 10.10.10.10
  username: cisco1
  password: cisco1
  commands:
    - show ip int brief
    - show ip route

- device_type: cisco_ios
  ip: 10.10.10.20
  username: cisco2
  password: cisco2
  commands:
    - show version
    - show ip vrf


- device_type: cisco_ios
  ip: 10.10.10.30
  username: cisco3
  password: cisco3
  commands:
    - show running-config
    - show ip ospf nei
```

Json(.JSON)


```
[
  {
    "device_type": "cisco_ios",
    "ip": "10.10.10.10",
    "username": "cisco1",
    "password": "cisco1",
    "commands": [
      "show ip route",
      "show version",
      "show bgp summary"
    ]
  },
  {
    "device_type": "cisco_ios",
    "ip": "10.10.10.20",
    "username": "cisco2",
    "password": "cisco2",
    "commands": [
      "show ip int brief"
    ]
  },
  {
    "device_type": "cisco_ios",
    "ip": "10.10.10.30",
    "username": "cisco3",
    "password": "cisco3",
    "commands": [
      "show int gig 0/0",
      "show ip vrf",
      "show mpls ldp nei"
    ]
  }
]
```

SSH Configuration on Devices

config t

ip domain-name shemy

username cisco privilege 15 password cisco

enable secret cisco 123

crypto key generate rsa encryption

##then type

2048

ip ssh version 2

line vty 0 4

login local

transport input ssh

IOS used for cisco (router and switch)

- ***For Router:*** c7200-adventerprisek9-mz.151-4.M.image
- ***For Switch:*** i86bi-linux-l2-ipbasek9-15.1g.bin
- To download IOS use below links:
  For Router: https://cios.dhitechnical.com/7200/
  For Switch: https://networkrare.com/download-cisco-iou-iol-images-gns3-gns3-iou-vm-oracle-virtual-box-l2-l3-cisco-switch-images/

Software

- GNS3, Version:2.2.40.1
- PyCharm, Version: 2023.2.4 (Community Edition)