University of Hertfordshire
School of Computer Science
MSc Computer Network and System Security

Final Project Report
7COM1039 – Computer Science Masters Project
23rd September 2022

Title
Network Automation Using Python Script (Network Programmability)

Student Name: Md Tahtihal Anher
Student ID: 19062288
Supervised by Honorable **Dr. Eric Chiejina FHEA**

Project Declaration

This report is delivered in potential fulfillment of the requirement for the degree of Master of science in Computer Network and System Security at the University of Hertfordshire (UH)

It is my act except where I demonstrate in the report.

I did not use any human involvement in my MSc project.

I have full consent and permission to use this project in university website as learning material or exploring knowledge.

Network Automation Using Python Script

Table of Content

List of Tables

List of Figure

Dedication

By the grace of Almighty Allah Most merciful and beneficent

My final project dedicated to my beloved parents Muktadir Ali and Hasna Begum, they always support me emotionally and financially with motivation and prayers. I am also gratitude to Honorable **Dr. Eric Chiejina** for supporting me consequently.

Acknowledgement

Alhamdulillah, I have accomplished my MSc project. Principally, I praise and acknowledge Allah (SWT), most beneficent and merciful. This report appears in its potential form due to assistance of various people. It provides me great amusement to express my gratitude to all selfless people who support me and encourage me to make possible.

Therefore, I would like to force out my admiration and specially thanks to my supervisor Prof. Eric Chiejina for his potential mentorship, guidance and motivation throughout the project. He made this project so exciting and attract me learn and implement new technology.

Finally, I would like to concede gratitude to my family member as it would not have been possible to accomplish my final project without their support and encouragement.

Abstract

This project arrangements are figured out by a plan, deploying automation environment and its utilization in real world network with its advantage.

The first segments of the project create a background and then investigates a range of probable approaches to automate network device using python programming language. It demonstrates two type of network automation in real world including the process of automation and establishing automation environment and its outcome.

According to this planning, each part of investigation expresses the summary of outcome and how python programming language can automate network devices, and how it is effectively beneficial for an industry or organization. This project utilized network emulator software “EVE-NG” to automate network devices, each part illustrates the outcomes of using python script in network device. In addition to thoughts on judgments made and their degree of correctness, the summary includes snippets of code written during the implementation phase.

After the summary of deployment, this project interprets how its outcome meet with research questions and how this investigation effectively significant in real world network management. It uses a number of methods to judge success, including qualitative analysis, automated testing, and user feedback.

NETWORK AUTOMATION USING PYTHON PROGRAMMING (NETWORK PROGRAMMABILITY)

1.0 Introduction

According to recent cisco research, 95% network tasks are monitored and configured manually. For the same tasks in deployment, several personnel must be employed. and it consumes lots of money, time and human involvement and it is the biggest issue for network managers is the enhance of IT cost of network operation and deployment. Manual operations are 2 to 3 times superior to the expense of the network industrially.

In heterogenous network manual deployment usually fetch human error and its troubleshooting is highly time consuming also need more worker to maintain the environment. Manual configuration also risky in various way, An incident that occurred in 1997, for instance, demonstrates how a Florida Internet Exchange complaint of a misconfiguration led to a period of interruption that spread across the internet.

Newly also a misconfiguration of BGP (Border Gateway Protocol) appeared in Pakistan and its result was two our blocking the entry of YouTube in worldwide.

Furthermore, this experiment aims to identify and demonstrate the best practice of network deployment that can progress the productivity of scripting in constituting the network devices and monitor to figure out the contrast in performance to network device configuration can be done manually or automatically.

In this demonstration two different topology explicit how scripting execute automation in realistic way that decrease the point in time for configuration with no error as well how decrease human involvement in same task.

These days, automation is becoming more and more popular due to its fantastic results and advantages, especially given the rising number of network devices. The definition of automatic is "doing or using in a way that is independent of outside impact and individual control." [2] Network automation offers businesses thousands of dollars in benefits; it permits the configuration of many devices in a short amount of time, eliminates the possibility of a misconfiguration caused by human mistake, which reduces operative expenses, advances security, increases productivity ratio, and, most significantly, is a method that is affordable, executes consistent results, and affords elasticity and flexibility. [1].

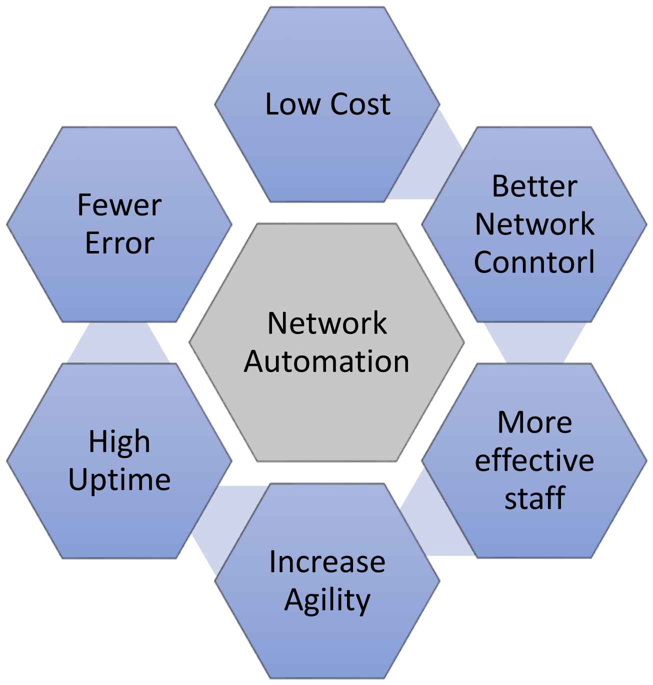


Figure 1.1 Various Advantage of Network Automation

This experiment interpret how network automation is effective with it is all attributes. The significance and applicability of this research will help network managers automate the deployment and setup of their networks.

Figure 1.1 exhibits attributes and facility of automation process including fewer errors, low cost, and fundamental network controlling. These three segments are significant in every company or industry also time savings. Network Automation is now applicable to all networks. Network automation is carried out by data centers, maintenance providers, and enterprises thanks to computer hardware- and computer software-based solutions that increase productivity, decrease human error, and save operational costs.

There have three types of platforms to deploy automation is highly recommended to get better performance – Enterprise Networking, Data Centre Networking and Service Providers. First Automation started by cisco in Enterprise Network, then Juniper, Palo Alto and Huawei also commenced. Not only automation processing in Enterprise and Traditional network, Nowadays Network Automation is implementing in RAN (Radio Access Network) for new 5G Network, in radio access network subscribers also huge, Automation is a key part of RAN along with Artificial Intelligence.

The purpose of this research is to show the most effective programming language for improving the effectiveness of scripting when configuring network devices using Python, as well as how different automation can influence and facilitate network automation qualities in heterogeneous networks.

1.2.0 Programming Language

To accomplish this project python programming language is utilized, there have lots of advantage in networking of python programming language such as its library can establish SSH connection by a tunnel from automation tools to device, also extremely easy to read and write. Another significant segment is big data, python able to ready and store big data by few lines in script. Six crucial advantages of python are

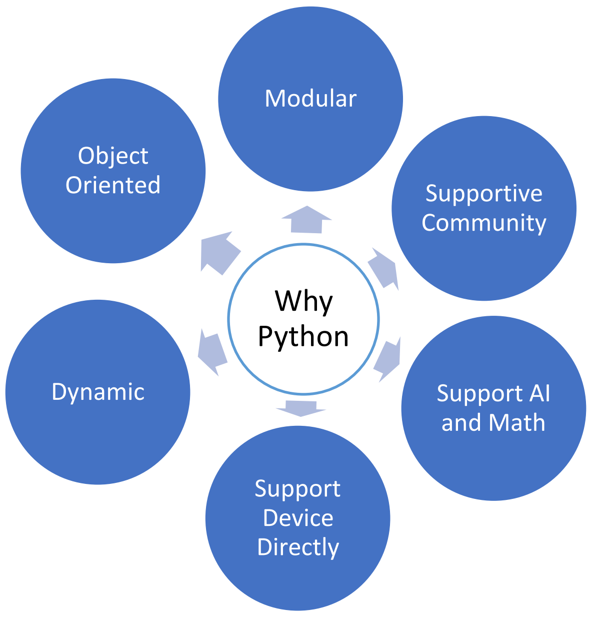


Figure 1.2 Advantage of Python Programming Language

1.3.0 Research Aim

The main purpose of this project is what is the best solution in heterogenous network which will be effectively beneficial for an organization or a company along with how network automation using python programming language can reduce human involvement and save time and these facts are a significant part in real world technological problem.

1.4.0 Objectives

In consequence my objectives of this experiment are:

- To identify a quick and effective solution for network configuration and troubleshooting by using programming language rather than manual command line.
- To create a dynamic and static data set for network device to utilize it by python programming language.
- To illustrate various network automation processes for real world network configuration problem.
- To point out the best automation process, which is more effective, fast and easy to troubleshoot in particular vendor.
- To evaluate the configuration by POSTMAN software to check performance and flaw.

1.5.0 Novelty

This experiment is practically exceptional than other project in network automation, all routers' data is stored in curly braces python dictionary directly rather than using any additional data format, therefore it is easy to call specific device from any place of script using just dictionary's name which is really significant in heterogenous network, that mean is unlimited devices able to store by dictionary and this process makes automation faster.

Another specialty is exemplified by this project, it explains two types of network automation process with providing proposed script and automation process along with troubleshooting, which will demonstrate the best type of network automation and which automations support particular device, which will solve the incertitude of an organization.

1.6.0 Legal and Ethical Issues

Although this project is demonstrating with comparatively no risk, however, there have some clarification. Emulator software is a free open-source platform and visual studio code by Microsoft. Maximum data sets are obtained dynamically by its own VM's cloud NAT, and some of data used statically, which are entirely free from human involvement or participation. As well as according to GDRP (General Data Protection Regulation), no data has been used in this project which is using by any company or organization.

1.7.0 Research Questions:

The Literature evaluation has been accomplished in considering with a focus on various research question that need to be demonstrated by this project. Along with network automation attribute and how significant in heterogenous network to provide most valuable facility with high security, these research questions will meet with the outcome of whole project, deployment, and testing of performance. This project about using programming language to automate network device to reduce time and human activity rather than using command line along with providing the answers to following questions:

1. Does Automation decrease human commitment which generates efficient and time saving resolutions for proper and dynamic protection of large heterogeneous network?
2. How Network Programmability fetch solution for dynamic remote network using secure shell (SSH), how secure SSH?
3. In heterogenous network, what is the most significant value of utilizing network automation?
4. Why python programming language in network automation? What is the key value and facility of using python programming in automation purpose with its library?
5. How many types of automation available to deploy in networking? And which automation is the most popular and reliable with latest technology?

Above these questions output will explain of entire project consequently. To Accomplish this project successfully and provide the appropriate acknowledgement of research questions, experiments topology will be in two segments. All section will illustrate automation process with python programming where every segment can meet with the research questions potentially.

1.8.0 Issues and Risks

Many networks engineer unwilling to use network automation in broken network, they use remediation tools for Command Line Interface. This issue happens only in broken network, but nowadays python have various way to detect the exact device where problem is occurred by latest mechanism. However, there are risks involved in adopting automation. Automation is no different from any other badly planned and executed procedure in that it has the potential to crash the network.

Another fact is Network elasticity, Network automation has fallen behind automation of computing and storage systems, and it has to catch up. Companies that put off adopting full IT automation run the risk of losing out to more agile rivals.

One example in network validation, pre and post change of network validation. If any pre-change check fails in network validation, then terminate the change. Similarly, notify network staff and maybe undo the change if a post-validation check is unsuccessful.

1.9.0 Security Issues

In real world automation tools such as solar wind or Cisco NSO, it is easy to encrypt username and password. In automating time plain text of username and password is highly risky , attacker can grab it from script by brute-force attack. For this issue it is mandatory to hide password and username by RSA encryption.

1.10.0 Commercial and Economic Aspects

According to ACG and Cisco Research , Network automation can reduce the cost of 55% by task. [1]

Three segments illustrate significantly for economic benefits :

1.Agility

2.Speed

3.Costs

Network automation has benefits for enterprises, but it also has risks. For starters, designing an automated solution will be more expensive the more complicated the system is. As a result, implementing an automated plan will cost money up front even though it is likely to be profitable in the long term.

Although automation process reduces the cost, But to setup automation environment it also costly. Skilled professional is mandatory for an organization to operate automation, otherwise consequent flaws make the cost higher which is a part of risk commercially. An error could spread quickly in a system that was not designed well , therefore network automation have to apply well organized platform with latest mechanism otherwise cost will be reversed.

This proposed project exemplify how network automation can reduce cost and human involvement with low risk practically with new experimented process of data storing, which is potentially beneficial for a company. Also, this investigation differentiates the best mechanism of network automation, where a company able to verify by its result which type automation is commercially effective and which device supports latest mechanism.

2.0 Literature Review

The use of Python programming techniques has become more accessible as numerous ways for network or system automation have advanced. One study's interpretation of the results from the improvement and testing analyses the prospect of creating network automation in Python programming that extends a variety of mechanization in a multivendor environment [3]. Trends in using the automation hypothesis are especially helpful for lessening time and manual engineers in computer network setup. Concurrent RANs (mobile networks), like those managed by mobile operators preparing to switch from 4G to 5G, would have to think about constructing a more standard-based, fully automated control, or software-based automated control, mobile network [19]. End-to-end quantum encryption is now potential thanks to the expansion of the virtual network approach and service automation [4]. Due to the substantial changes in the network and system automation environment over the past few years, this has transpired. Innovative solutions are required for new challenges, and infrastructure must be implemented in a way that adapts dynamically to new frameworks. This dictates several automations, including secure automation [8] Network virtualization and additional novel network models, such (SDN) Software Defined Network, have shown to be flexible in terms of network system management and services[13].

Operating network has been progressed regularly when necessitate and it is fundamental because the importunity of network is enhanced on exact time. QoS (Quality of Service) of Network automation, dynamic architecture, and control has been executed [10].

As the medium for supervision and data ownership networks, establishing a trustworthy communication network is a difficult sector. In order to design a network, there may be a number of architectural obstacles, such as whole-traffic, compact node arrangements, bandwidth restrictions, packet retransmission, intervals, and drops. In addition to a precise hardware layout, firmware mechanisms, and the most recent application-layer defective-node-penetrate-algorithm, an inquiry has executed a trustworthy HTTP based mechanization network of huge, interlinked microcontroller-based points. For a magnificently operating cryogenic regulator network of superconducting rectilinear accelerator at inter-university accelerator center, New Delhi, India, the verified system is dispersed in an isolated Local Area Network (LAN) of more than 50 surrounded servers, each of which is built out of on-chip Transmission Control Protocol/Internet Protocol (TCP/IP) loads executed on Advanced RISC Machines (ARM) workstations as nodes [7].

Automation was attained through individual utilizes of a coding language, in example python language As a result of its simplicity compared to java and C, Python has emerged as the most popular programming language for automation. Python was intended in 2018, 2007 and 2010 as the most excellent programming language of the year[17]. Since its initial release in 1991, Python has been utilized for server-side web improvement, software development, mathematics, system programming, and other purposes. [4] Python has various benefits all over more coding languages, python acts on several programmed for example Mac, Windows, Raspberry Pi, Linux, and so. Raspberry Pi is one of most significant use for emulation, maximum lab deployment and simulation usually do by Raspberry Pi. In Network Automation sometimes need to deploy whole architecture in simulation mode such Cisco CML Lab, if in mean time need to run python scripts for automation purpose Raspberry Pi can fetch a significant impact on it.

It features a simple syntax that grants creators to design programs quickly with less lines of code and deploy them faster than with other coding languages. Its modest composition is similar to that of English [6].

The study shows novel ways to configure network devices like Cisco, Juniper, mobile networks, and others by utilizing automation and Python programming, which cuts down on configuration time for actual equipment and makes maintenance easier [9].

The organization enhances network protection by identifying and securing vulnerabilities, which helps to create a stable network. Similarly, research has shown that utilizing the Python programming language to automate the development procedure for remote network devices is time-effective; the practice of advance network devices can be time-exhausting, as the operator must individually upgrade every single device under monitor [5].

Due to their significance and the purpose of numerous practices, research into the automation of traditional networks has expanded dramatically in recent times. One study illustrated the shortcomings of standard network management and try to resolve them by operating SDN [19] A new automatic policy is defined that is mostly applied to modernize remote network devices employing software development, also known as software illustrations.

Another technique for automating network design, static and dynamic routing, designing VLANs, backing up and renovating data, and additional network administration projects has been established as a web-based application operating Python programming [21].

While the As-RaD System, an additional network automation paradigm, has been proposed. It was conceived with Python and the Django framework and employs a REST API architecture [26]. This As-RaD System outperforms both the NAPALM method and the Paramiko approach by 92%. Some of administrator use Netmiko and some of Paramiko. But all of these are CLI based automation. Nowadays technology robustly growing up, so CLI based automation also going back, REST API & Netconf is the one superior automation in Networking currently. In Paramiko and Netmiko there have some difference is runtime, paramiko is little bit faster than netmiko [29].

Another most popular and heavy used network automation platform is “Ansible” can configure various devices such MikroTik, cisco and arista routers by using raspberry pi to configure routing protocol mostly[27] and the user manager's implementation. Based on Python libraries like Netmiko, one of the inventors designed an appliance known as the Network Management Platform (NMP) [24]. As a result, configuring more than 10 Cisco devices automatically needs only 2% of the time of performing so manually.

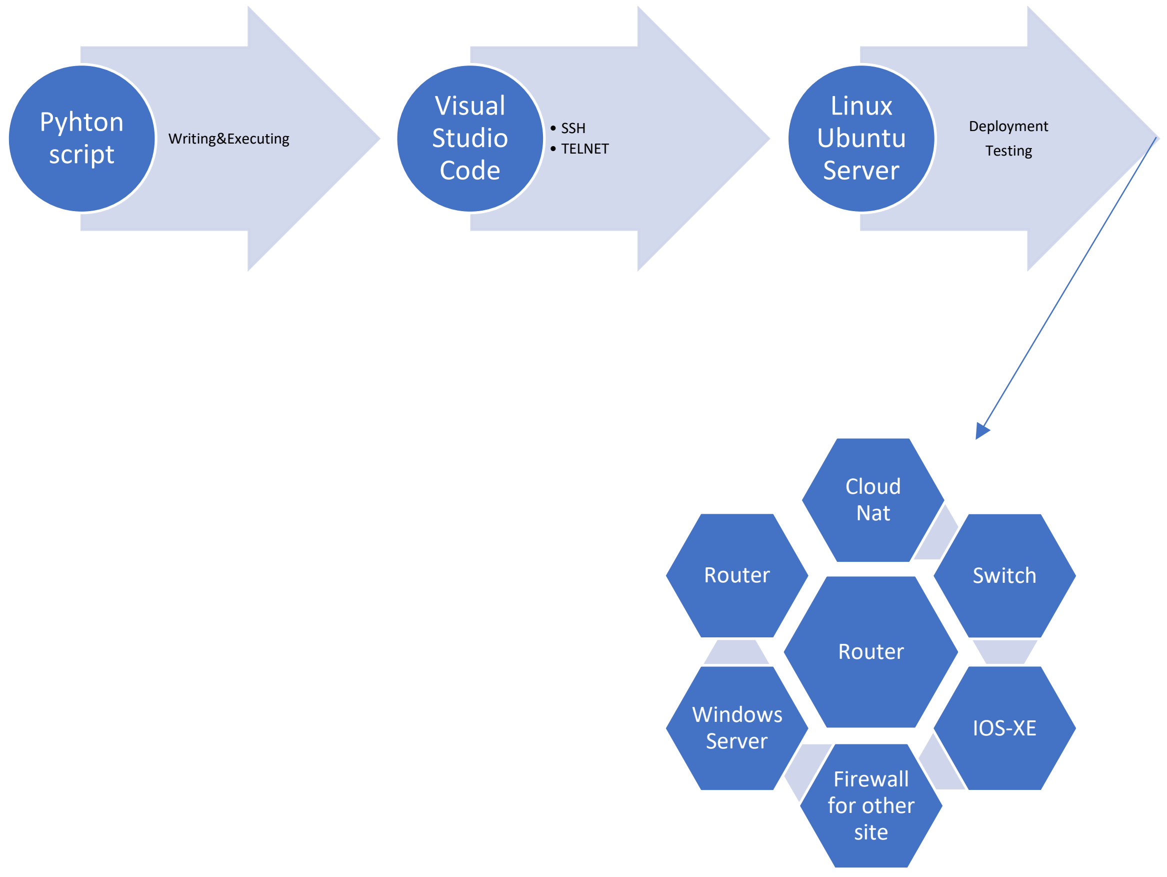


Figure 1.2 the process of Automation

Above process interpret how network autoamtion work practically with writing and excuting code , when code excuted it needs SSH plugin with internet connectivity for remote access by Linux Ubuntu Server. In this instance, maximum vendor currently use one of the mos popular automation tools known as NSO (Network Sevices Orchestrator) powerd by Cisco.Using this tools it is very easy to push script ot destinated IP address to configure or troublehoot the devices, for this tools also mandatory SSH authentication [11].

Along with above automation this experiment will also show REST API automation by HTTP request.

These automations are latest technology in networking, where network administrator able to use python script to automate devices such as cisco by sending URL request. For this Automation maximum time requests library use for this purpose [21] and it is more dependable and lightweight. In this case device must support YANG data model because only for IETF (Internet Engineering Taskforce) interface, when http request sends to configure devices that time URL communicate with IETF interface in the first instance. Also need to create and assign in python script about IETF interface of device and what type of data format will be replied after calling by HTTP request such Either XML or JSON [26].

Overall, two of automation will be demonstrate in this experiment along with it is all attribute where it can meet with research questions and finally will implement which Network Automation will be more effective with time savings and can get output very easily along with deploying python scripts. Altogether three python libraries will be used heavily Request, Netmiko and Ncclient in their separate platform to automate certain devices such as cisco, however nowadays maximum vendor trying to develop and align their products in top list of higher technology such as Juniper, Huawei and MikroTik [31]

2.1 Critical Analysis

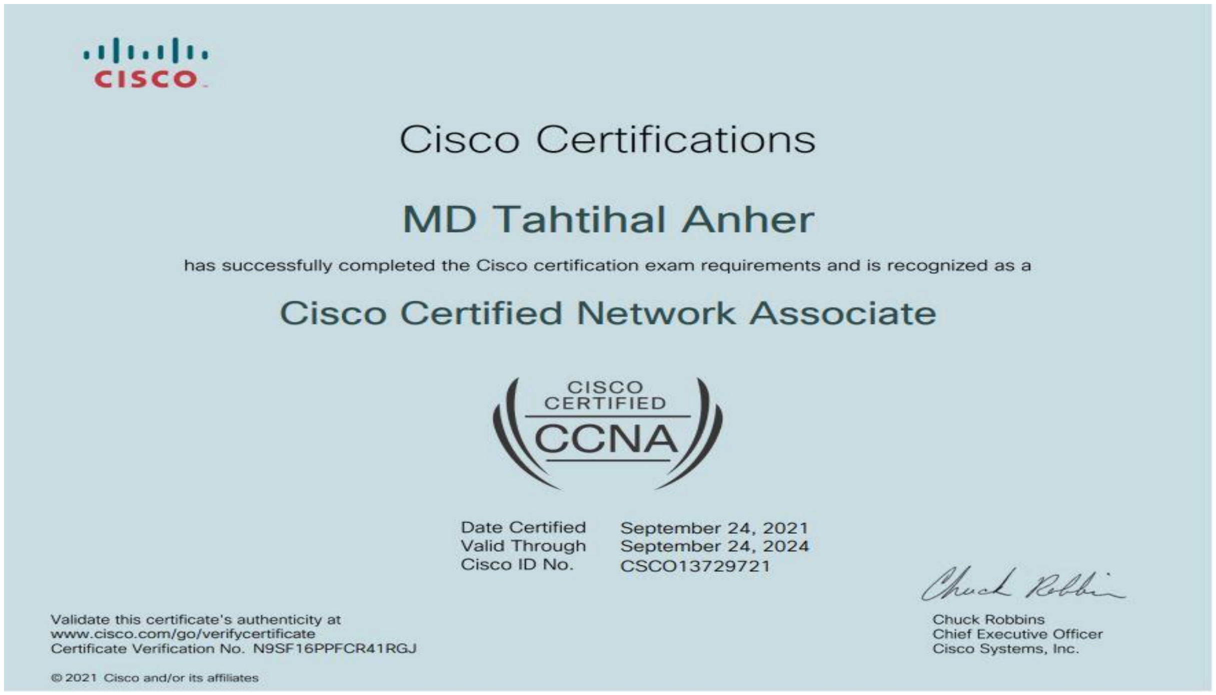
The proposed investigation expresses the best way to store data using python programming for network automation purpose and how this technique is time savings and make automation faster where it is really beneficial industrially.

This project also interprets two type of network automation, CLI based network automation and HTTP request-based network automation including its distinction along with what is the best way of network automation in real world. This difference is highly recommended for an organization to setup own server or DNA center to purchase network devices such as routers. Lots of new device still not supporting latest technology such as HTTP based network automation. This experiment demonstrates practically the best network automation process and which device supports latest mechanism; this fact is potentially beneficial for an organization.

3.0.0 Methodology

This section will describe the methodology of the project and the procedure of programming related tasks, tools, and technique used such as data assigning, data processing, topology, automation procedure, SSH Connection, DHCP Setup to obtain IP address from home router and VISUAL STUDIO CODE as source-code editor.

To make network topology there has been used Emulator Software “EVE-NG” and to run this software must need Virtual Machine, in here VMware workstation pro (paid version) has been used. Another section will explain the requirement and its configuration. This section “Methodology” is set down by following research questions. Programming language pythons is utilized in this project. It is entirely networking project, so fundamental networking concept is significant, I accomplished a course CCNA 200-301 (Cisco Certification) and found a course in Udemy which is python programming for network technologist (python 3) by David Bombal. Figure (4.1 & 4.2) shows my certifications to successfully finish master’s project. And, another part is Data Set, it is one of crucial part of project. IP addresses used from DHCP Server from my home router, and it is getting from NAT (Network Access Translation), When API is called topology must connected with cloud NAT, and cloud NAT distribute IP address to DHCP client from home router.



Cisco Certifications

MD Tahtihal Anher

has successfully completed the Cisco certification exam requirements and is recognized as a

Cisco Certified Network Associate

CISCO CERTIFIED CCNA

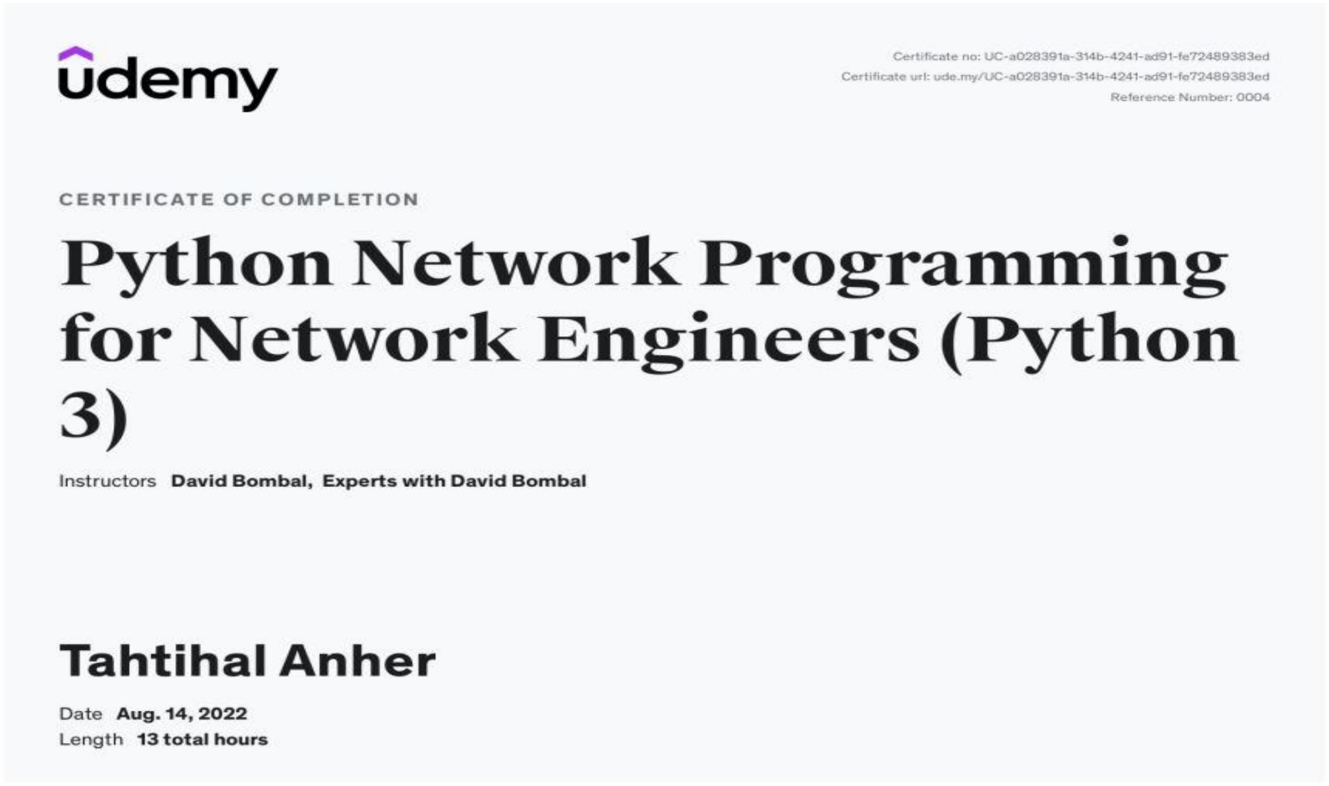
Date Certified	September 24, 2021
Valid Through	September 24, 2024
Cisco ID No.	CSCO13729721

Validate this certificate's authenticity at
www.cisco.com/go/verifycertificate
Certificate Verification No. N9SF16PPFCR41RGJ

© 2021 Cisco and/or its affiliates

[illegible]
Chuck Robbins
Chief Executive Officer
Cisco Systems, Inc.

Figure 4.1 Cisco Certified Network Associate Certification



Certificate no: UC-a028391a-314b-4241-ad91-fe72489383ed
Certificate url: ude.my/UC-a028391a-314b-4241-ad91-fe72489383ed
Reference Number: 0004

CERTIFICATE OF COMPLETION

Python Network Programming for Network Engineers (Python 3)

Instructors **David Bombal, Experts with David Bombal**

Tahtihal Anher

Date **Aug. 14, 2022**
Length **13 total hours**

Figure 4.2 Python Certification for Network Automation

3.0.1 Justification

It is really important for this project to set up a network automation environment and the methodology to demonstrate the investigation and to meet with the research questions.

Device software has to be imported by authentic vendor, I choice Cisco devices (Virtual Images) and this authentic vendor supports EVE-NG and GNS3 , in this project I used EVE-NG network emulator to automate virtual images which images designed exactly same as real device by Cisco for study purpose and it support virtualization.

According to the main goal of this project python programming language is a crucial part, for this aspect I used “Microsoft Visual Studio Code” which is open-source platform and also supports “SSH Version 2” connectivity for Cisco images.

This methodology selected for 4 reasons:

1. Virtual Device Images (Exactly same as real device)
2. Virtualization (For Network Emulator)
3. Programming Language (For Automation Script – python)
4. Visual Studio Code (For SSH Version 2 connectivity)

3.1.0 Emulator Overview

Resources for forming connections between virtual devices and real and more virtual devices are offered by the Emulated Virtual Environment-Next Generation (EVE-NG). It is feasible to dramatically shorten the utility, reusability, management, interconnection, and allocation of EVE-NG features. As a result, EVE-NG save time and money. The clientless network emulator known as EVE-NG extends a browser-based graphical user interface (GUI). Users can put together network nodes from libraries and downloadable templates. Requirements: CPU: Intel CPU with Intel® VT-x and EPT capacity for virtualization OS : VMware Workstation pro-16.0, or later Desktop Linux, Windows 7, 8,10,11 The bare minimum hardware required for laptops and computers is indicated in the table:

Process	Estimation
CPU	Intel i5/i7/ryzen 5/ryzen 7 (Minimum 4 logical processor) Must be enabled virtualization in Bios.
RAM	8GB
HDD/SSD SPACE	40GB
NETWORK	LAN/WLAN | NAT/BRIDGE

Table 4.1 about Minimum Requirement of System

3.2.0 License Agreement of EVE-NG

In this project I used Emulated Virtual Environment – Next Generation (EVE-NG) community edition v5.0.1-12 as my network emulator, Community edition is free for learning. EVE-NG community reveal this community edition for study purpose without any cost, there have also another edition it is EVE-NG pro, which is paid version.

3.3.0 Types of Automation

After huge research, its expressed that total three type of automation in Networking possible by python programming, but it depends on devices software like cisco have various software such cisco IOS, IOU, IOS-XE and so on. Three types of automation which are tested in this project by separate python script:

Automation Type	Supported Device
CLI Based Automation	**Arista Veos, Cisco ASA, Cisco XR, HP Comware7, Juniper Junos**
RESTCONF	**IOS-XE**
NETCONF	**NX-OS, IOS-XE, IOS-XR**

Table 4.2 Supported Device by automation type

3.4.0 Specific Progression

Experiment is explained by three segments in order to interpret three types of automation according to Table 4.2. Every segment has their own topology along with own python script. Data set is different due to different vendors device. All data set is assigned dynamically without few devices. To provide high security SSH connection authentication is static. Data set of secure connection is expressed in separate table for separate topology. Moreover, various python library used in this project, all libraries overview and purpose are described practically.

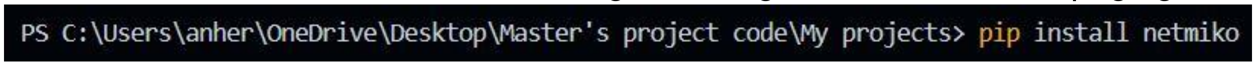
3.4.1 CLI Based Automation

To implement CLI based automation "Netmiko" python library is utilized in this project. A Python module known as netmiko, which maintains many SSH vendors, creates it simpler to link up to network devices utilizing SSH. Paramiko, the default SSH library in Python, is superior with vendor-specific logic by this module.

Installation of netmiko:

To install netmiko library "pip install netmiko" command fetch netmiko library in python to deploy. There have lots of key features in Netmiko Library:

1. Establish SSH Connection.
2. Recognize the command prompt string for the device and vendor provided.
3. In order to determine whether the device has finished transmitting its answer, issue the command and then carry out the necessary string handling.
4. Take care of the different string handling details, such as paging and



```
PS C:\Users\anher\OneDrive\Desktop\Master's project code\My projects> pip install netmiko
```

terminal widths.

According to cisco research also have another library available for CLI based automation but this library utilizes for network device testing to get the output or running configuration known as "Napalm". This library usually uses for getting the output of BGP (Border Gateway Protocol) routing protocol, the output exhibit as JSON (JavaScript object Notation) also user able to define which type of data will come as output, usually JSON is a good format to read and monitor.

3.4.2.0 TOPOLOGY FOR CLI BASED NETWORK AUTOMATION

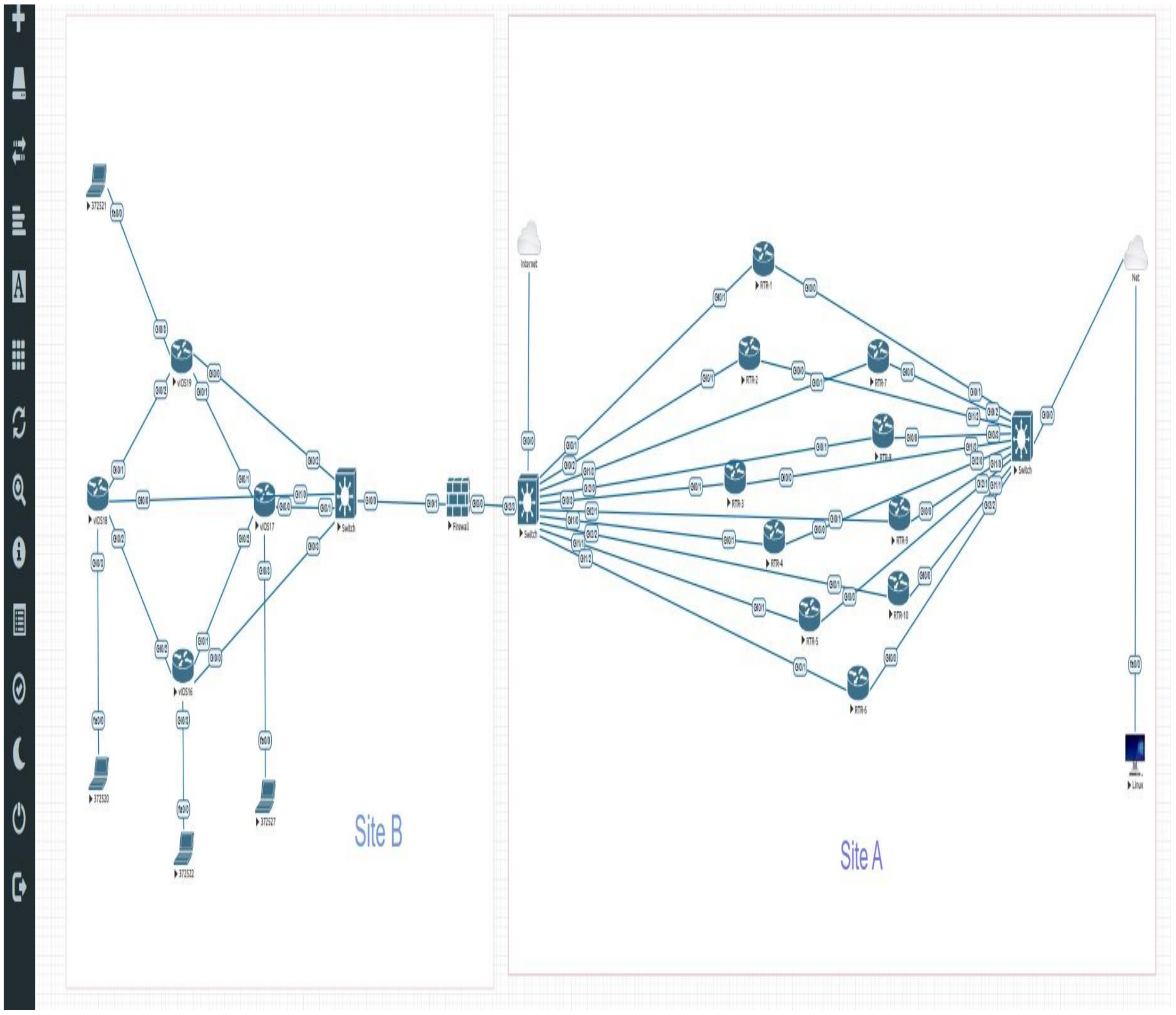


Figure 4.3 show the topology of CLI Based Network Automation

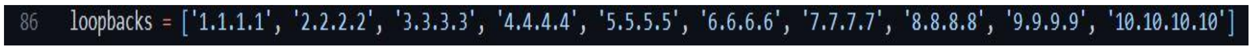
In above topology there have two sites Site B and Site A, initially Site A have 10 routers. All routers are Cisco IOS version which supports only CLI based Network automation. This topology created in EVE-NG community edition which is free version for learning opportunity provided by EVE-NG community. All routers have their own credential for authentication to get access in virtual environment. All devices configured by python programming with establishing SSH connection. Data Sets are assigned in another segment.

3.4.2.1 Topology Overview – Design Traditional Network and Routing Protocol - OSPF

With the aid of (EVE-NG), a large-scale conventional network is constructed, and a server is created and connected with this network.

The layout of traditional network contains leading routers, switches (Switch-1 & Switch-2) bother switch are layer 3 switch. The main router, which is considered as an access to the internet and the network, is linked to the automation server. Other routers considered as branch routers and tunnel routers. Each branch has own terminal router, the summation of total routers is 14 and altogether 25 devices are in this traditional network. This strategy manipulated as a genuine enterprise network. Figure 4.3 shows the topology of traditional network.

Initially Site B is fully configured in this topology, Site A had explained in first segment. All devices of Site B able to reach all router to the automation server. For every router has their own "loopback" interface to monitor and test purpose. Management interface is mandatory to configure for scripting purpose, all devices have separate management interface with IP addresses, without management interface it is entirely unable to deploy automation script to router.



```
loopbacks = ['1.1.1.1', '2.2.2.2', '3.3.3.3', '4.4.4.4', '5.5.5.5', '6.6.6.6', '7.7.7.7', '8.8.8.8', '9.9.9.9', '10.10.10.10']
```

Above python set shows the loopback address of all routers, this loopback address is configured to routers by automation.

On the other hand, in this topology Routing protocol OSPF stands for "Open Shortest Path First" is used, where OSPF use Link State Routing (LSR) algorithm and operating with a single autonomous system (AS). The most convenient feature of OSPF is it gathers Link State Information from other router which routers are available and comprise a topology map of the network. The topology is illustrated as a routing table for Layer 3 packets to the internet layer by their destination IP address. OSPF Support Internet Protocol Version 4 (IPv4) and (IPv6) and support CIDR (Classless Inter-domain Routing). In Large Enterprise Network, OSPF widely used. In this project routing OSPF routing protocol has been utilized, where it has more flexibility and attribute than EIGRP and RIP Version 2. In addition, OSPF is an Interior Gateway Protocol (IGP) for transporting IP (Internet Protocol) packets inside a single routing domain, in example an autonomous system.

There have lots of benefit of using Open Shortest Path First (OSPF) routing protocol over other routing protocol such as RIP, EIGRP. 5 most valuable benefit of OSPF explained below:

- ✓ Faster Convergence [Network coverages faster for flooding routing changes immediately]

- ✓ Support For VLSM [For Heterogenous Network]
- ✓ Network Reachability [RIP routing protocol has limitation of 15 hops, where OSPF has not limitation]
- ✓ Metric [OSPF utilizes ‘cost’ metric to select best path]
- ✓ Efficiency [Send updates if any change in network]

In this project OSPF routing protocol is utilized due to its numerous conveniences and according to research questions need to demonstrate the heterogenous network automation where OSPF routing protocol is widely used in Enterprise Large-Scale Network. In this experiment will demonstrate using “Netmiko” python library how to configure OSPF routing protocol in automation process where traditional network configuration such as by command line configuration takes more times and people for more router and it is really expensive for network manager or company.

3.4.2.2 Main Goal of CLI Based network Automation

According to research questions this part also done successfully with its all attribute. In this experiment it is demonstrated that how python programming language can configure 14 routers within 1-2 minutes, where traditional configuration takes huge time, and more people require to accomplish it. But CLI Based python Network automation can configure “OSPF” routing protocol and other configuration in a same time by only one automation engineer, which is more beneficial and reliable for concurrent industrial challenge with proper authentication.

3.4.2.3 Data Sets

To provide device security, authentication & recognition there have specific data for specific devices. All devices have their own IP address and own authentication to get access in devices for configuration and monitoring.

Device Name	Username	Password	IP Address	Device Position
R1	anher	19062288	192.168.231.209	Tunnel Router
R2	anher	19062288	192.168.231.226	Tunnel Router
R3	anher	19062288	192.168.231.202	Tunnel Router
R4	anher	19062288	192.168.231.214	Tunnel Router
R5	anher	19062288	192.168.231.211	Tunnel Router
R6	anher	19062288	192.168.231.199	Normal
R7	anher	19062288	192.168.231.207	Normal
R8	anher	19062288	192.168.231.201	Normal
R9	anher	19062288	192.168.231.205	Normal
R10	anher	19062288	192.168.231204	Normal

Table 4.3 shows the data sets of Network Topology

Above table illustrates data sets of CLI based Network Automation Topology. All IP addresses assigned dynamically and to provide high security for authentication SSH credential is utilized manually. Total five routers acting as tunnel router R1, R2. R3, R4, R5. Before establishing the connection, data must be in destinated router such as IP address, Username, Password and all data must be assigned in python script before pushing, otherwise SSH connection will be refused. A flow chart will demonstrate the process of full automation procedure. However next section will explain about the data processing and how IP address and SSH credential has been assigned.

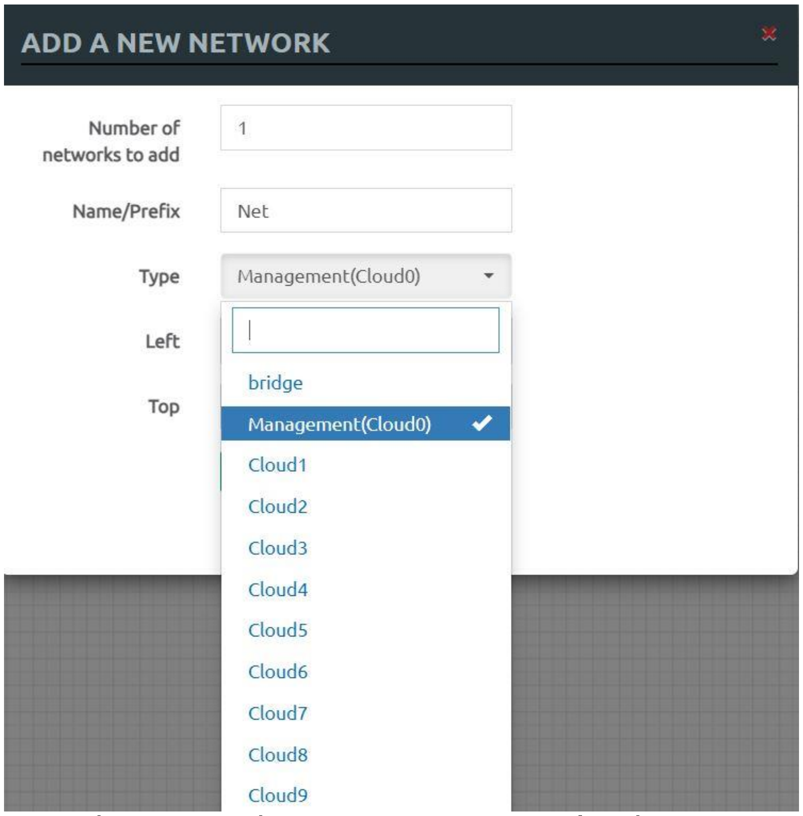
3.4.3.0 Data Collection

Table 4.3 has explained about all data sets of network configuration. IP addresses has been assigned dynamically by DHCP Server.

Two System Setting is Mandatory for data collection:

According to system settings Virtual Machine is mandatory to run EVE-NG. After running EVE-NG emulator there have two options for network adapter one is Bridge connection and another one is NAT. Bridge connection for LAN (Local Area Network) which computer or laptop directly connected via cable like RJ45 port and NAT is for Laptop or Wireless device which device using wi-fi for internet connectivity. In my project I am using Laptop and it is connected through wi-fi.

Moreover, in Virtual machine system setting for this project has given NAT, for this reason when turn on any device in GUI of EVE-NG and if need internet connection of this device it must have to be added with cloud NAT (Management cloud) of Network otherwise device would not get internet connection.


ADD A NEW NETWORK

Number of networks to add: 1

Name/Prefix: Net

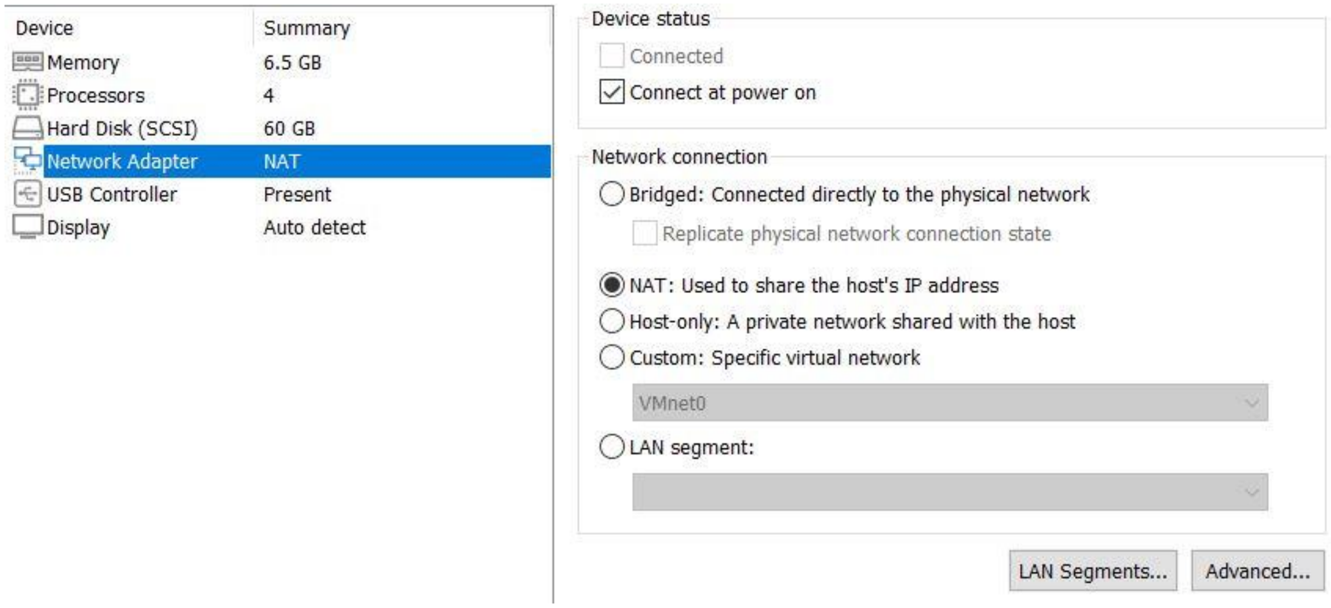
Type: Management(Cloud0)

Left

Top

Figure 4.4 shows Management cloud

According to Figure 4.4 to get internet connectivity in devices and collect IP address dynamically this configuration is mandatory. Also need another setting in Virtual machine changing network adapter to NAT.



Device	Summary
Memory	6.5 GB
Processors	4
Hard Disk (SCSI)	60 GB
Network Adapter	NAT
USB Controller	Present
Display	Auto detect

Device status

[ ] Connected
[✓] Connect at power on

Network connection

( ) Bridged: Connected directly to the physical network
[ ] Replicate physical network connection state
(●) NAT: Used to share the host's IP address
( ) Host-only: A private network shared with the host
( ) Custom: Specific virtual network
VMnet0
( ) LAN segment:

Figure 4.5 shows the Network Adapter Setting

Above two system settings are required to get IP address dynamically from Virtual machine DHCP Server. Along with those configuration DHCP must be enabled in device to get IP address dynamically. First, assigned a Management IP Interface and then DHCP is enabled in management interface to obtain IP address. One diagram will show about the setup of Management Interface and its command line. That configuration is done manually.

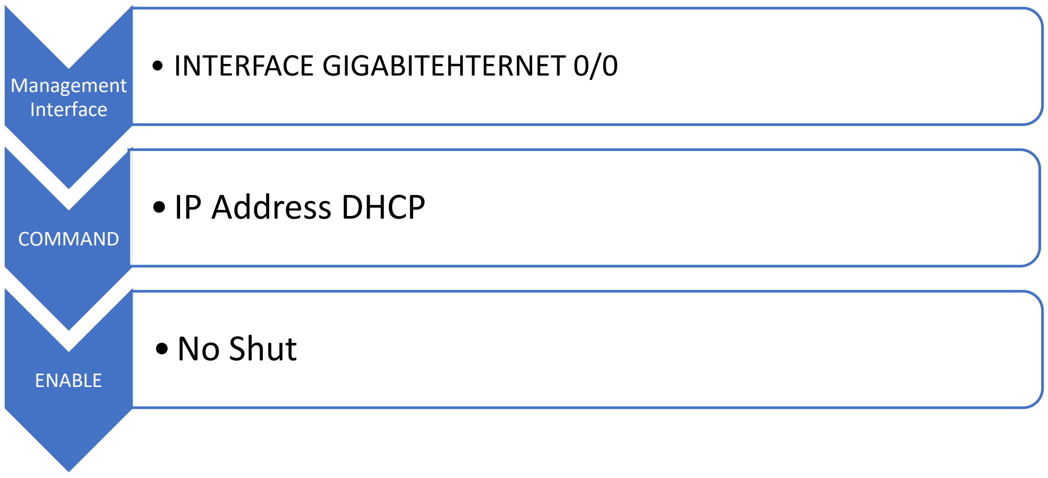


Figure 4.6 shows about Management Interface Setting

In this project this configuration used in every device, *Interface GigabitEthernet 0/0* is used as management interface in every router and this management interface getting IP dynamically by enabling DHCP client command “IP ADDRESS DHCP”.

This process only for remarking the IP address of routers to automate by python script. Before Automation process SSH setup is accomplished for security purpose manually. Otherwise, attacker/hacker can easily push their script to hack it, Secure Shell is significantly beneficial for security and authentication.

SSH Configuration commands:

- ✓ Hostname [Need to assign the hostname such as R1 for this project]
- ✓ ip domain-name [Need a domain name such as networkengineerpro.com]
- ✓ crypto keys generate rsa modulus 1024/4096 [To store RSA key for encryption]
- ✓ username anher privilege 15 secret 19062288 [Login credential]
- ✓ line vty 0 15
- ✓ input local
- ✓ transport input ssh
- ✓ ip ssh version 2

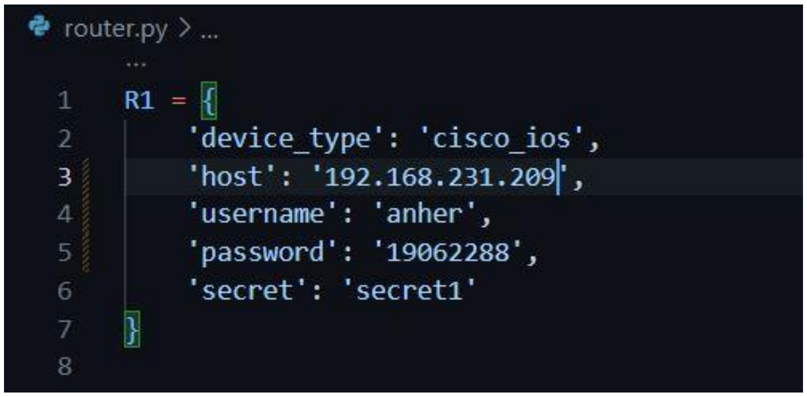
Above commands make a Secure Shell (SSH) connection to authenticate device. Even if anyone get the device with logged in, unable to get the password because it will be encrypted by RSA and privilege 15 means encrypted letter is 15 various characters.

In this project all devices have their own SSH authentication for automation purpose, these processes also demonstrate to keep remember of research questions. If compared to telnet, telnet does not have more power to protect device such as RSA which key used for encryption of DES or AES data key.

To establish SSH authentication here used “anher” as username and password is “19062288” [My name and Student ID] in every device to make implementation easier.

3.4.4.0 Data Store

To implement automation for configuration purpose data store is a significant part after assigning and collection of data set from the device. Data stores must be flexible for automation purpose where able to call from anywhere of script. There has various way available in python to store data such as using data format in example XML, YAML, JSON also using python dictionary curly braces are used to define data structure called dictionary. In first part of this project YAML data format has been used. YAML data format is flexible to call or open to process data in python programming. Using dictionary also adaptable to call the dictionary from anywhere of script. In this configuration curly braces dictionary applied to store data and credential of devices.

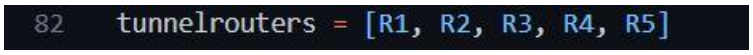


```
router.py > ...
R1 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.209',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}
```

Figure 4.7 shows curly braces dictionary

Above Figure 4.7 demonstrates a curly braces python dictionary to store data. In this dictionary R1 (Router 1) data stored such as ‘device_type’: ‘cisco_ios’, ‘host/IP’: ‘192.168.231.209’[device recognition] and very significant data are “username” and “password” which is the authentication of SSH (Secure Shell). So, this curly brace dictionary contain data of one device and this dictionary stores in a sperate python file where all curly braces dictionary stores for other device. When automation is required for Router 1 (R1), in python script R1 dictionary is called in order to recognize script or configuration will configure this certain devices R1. In this python file name as routers.py [Where All Dictionary stored], 10 dictionary stored, and every dictionary contain each routers IP address and credential. It is possible to add unlimited routers credential including SSH authentication by creating unlimited curly braces dictionary. When require automating or pushing script to certain device just need to call certain dictionary.

According to research questions in this project here utilized 10 routers in site B to automate full configuration by python programming instead of traditional command line which takes more time and more people. 10 curly brace dictionary is utilized here for 10 separate routers, also used from dictionary as a set for router separation such as tunnel routers:



```
tunnelrouters = [R1, R2, R3, R4, R5]
```

Those routers are tunnel router and it stored as a set-in python file, Where the set name is tunnelrouters and its value is R1, R2, R3, R4, R5. In python script these 5 routers are utilized to configure such as ROAS. Therefore, when need these routers to configure or automate by python script in main function or in condition, declaring only sets name “tunnelrouters” is enough to configure device by automation.

3.4.5.0 Installation of Automation Server

Automation server is the significant key part of this investigation. The entire automation process will run through it such as SSH connectivity, Script Writing and deploying, data processing, data pushing. Linux-Ubuntu is the OS that has been selected for the server. Basically EVE-NG server which is run by VMware workstation pro it is Linux based server [19]. There have some commands for this have been deployed in Linux Server.

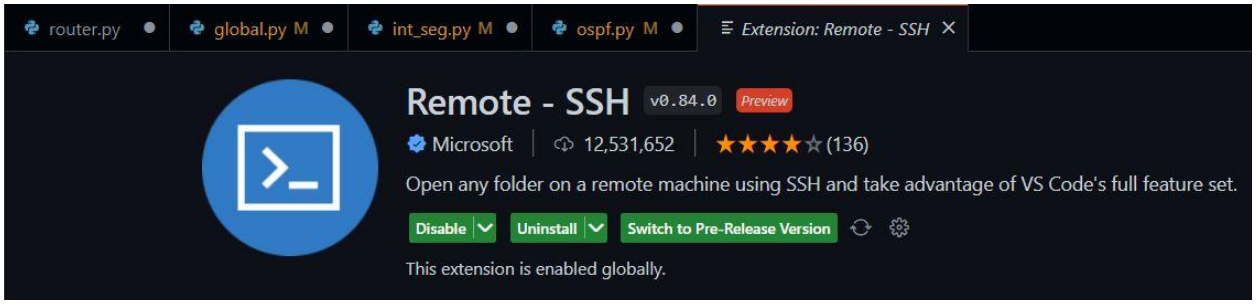
- ✓ apt-get update [System Update]
- ✓ apt-get install python3 [Automation Script is python, so python is mandatory]
- ✓ apt-get install python3-pip
- ✓ install pip3 [For Libraries such as Netmiko or Napalm]

Above all commands is successfully done in server for confirmation of python and its libraries where server can understand of python programming language and Netmiko library for CLI based automation.

When the automation scripts are applied, the server will be prepared. Python scripts for the automation were created using "Visual Studio Code". Direct SSH connections to the Ubuntu server are possible with VS Code. The server houses all of ACS's scripts.

Visual Studio Code is a Microsoft product also identified as VS Code, Source code editors for Windows, Linux, and macOS are available. Lots of feature included such a support correcting, syntax feature, intellect code generalizing and accomplishment, refactoring and entrenched version control system such as git.

There have also another potential feature included which feature used in this project, utilize all of VS Code's features by opening a remote folder on any distant system, virtual machine, or container with the Visual Studio Code Remote - SSH plugin installed and also able to work with files and folders anywhere on the remote filesystem once its connected to a server. In this project to establish SSH connection VS Code (Visual Studio Code) has been applied but authentication is mandatory. VS code requires a plugin for remote SSH connection.



Open any folder on a remote machine using SSH and take advantage of VS Code's full feature set.

This extension is enabled globally.

Figure 4.8 SSH Extension for VS Code

Figure 4.8 shows the extension which is mandatory for remote SSH connection to Virtual Machine Such as EVE-NG or direct Server. Establishing a SSH connection a SSH Tunnel automatically generates between Local OS and remote machine or Virtual Machine such as VMware workstation pro. Source code, Terminal Process, Running Application, Debugger all are deployed after establishing the SSH tunnel between local OS and Virtual Machine.

According to my research question and project plan working on network automation by python programming, therefore python extension have to be in my Visual Studio Code including which library is used int this project.

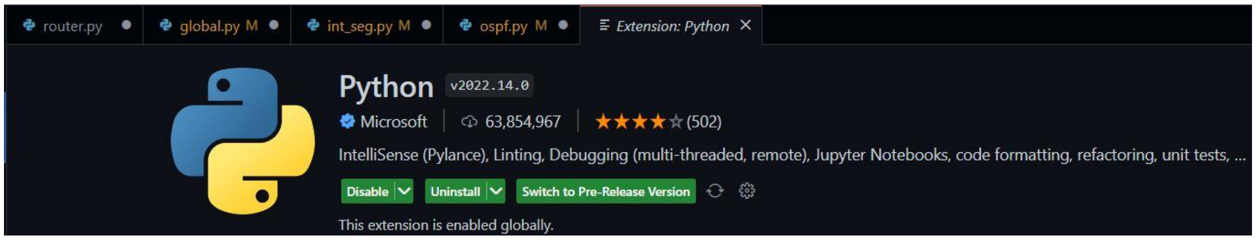
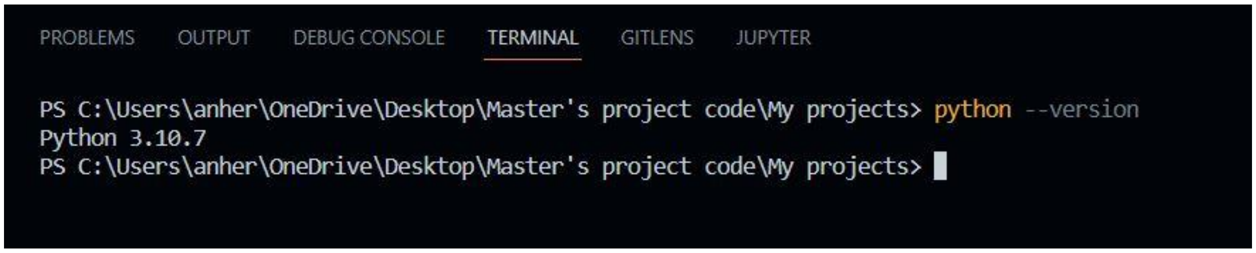

router.py
global.py M
int_seg.py M
ospf.py M
Extension: Python
Python v2022.14.0
Microsoft
63,854,967
★★★★☆ (502)
IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, code formatting, refactoring, unit tests, ...
Disable
Uninstall
Switch to Pre-Release Version
This extension is enabled globally.

Figure 4.9 python Extension for VS Code

Figure 4.9 shows the installed extension of python programming in Visual Studio Code to generate code. Code outcome is error without installing this extension where VS code will call for python in run time. After installing this extension also need to check is it installed properly by Visual Studio code terminal. In this project Visual Studio Code terminal used often to run script for automation purpose.



```
PS C:\Users\anher\OneDrive\Desktop\Master's project code\My projects> python --version
Python 3.10.7
PS C:\Users\anher\OneDrive\Desktop\Master's project code\My projects>
```

Figure 4.10 about python Version

After installing python extension in VS code, it is necessary to check python version to identify whether python installed successfully.

- ✓ Python –Version / python –v

After providing above command, if get the output the version of python such as here obtained “python 3.10.7”, that is mean that python installed in Visual Studio successfully and Visual Studio is ready for code writing and deploying.

After installing SSH extension for SSH connection and python extension to understand python programming language will be used in Visual studio code, automation platform is ready to use. But in meantime need to install require python libraries in Visual Studio Code otherwise after calling the library in script it shows error, in this project for CLI based automation I used 'Netmiko python library' to configure cisco devices also if need to add any data format such as in IPR I utilized YAML data format to store data such as hostname, host ID, username, password , Therefore in Visual Studio code must be installed or declared about data type of certain data or it is easy to import by calling commencing of script such as "import yaml /import json".

3.4.6.0 Flow Chart of Proposed Script

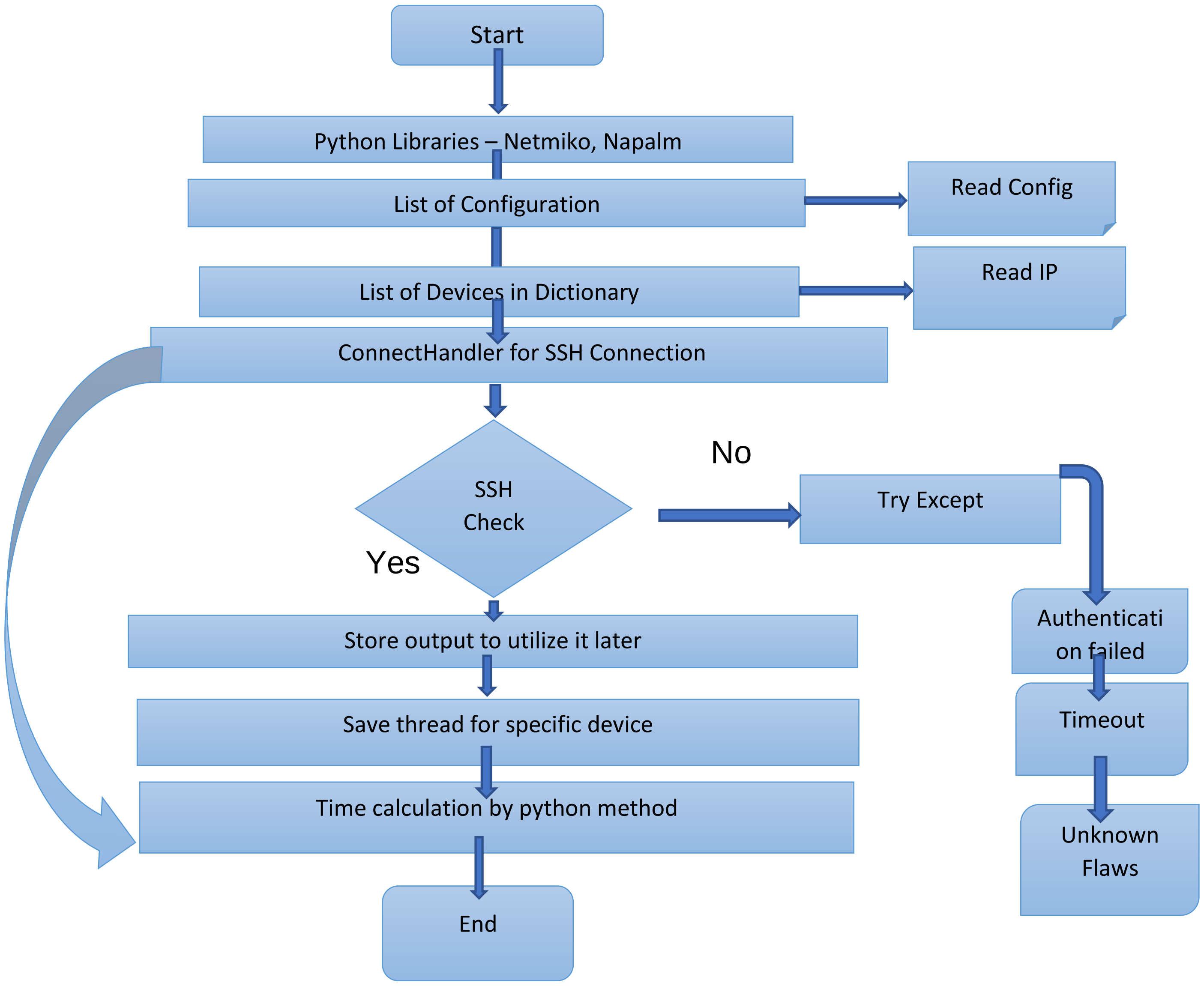

Start
Python Libraries – Netmiko, Napalm
List of Configuration
Read Config
List of Devices in Dictionary
Read IP
ConnectHandler for SSH Connection
SSH Check
No
Try Except
Yes
Store output to utilize it later
Authentication failed
Save thread for specific device
Timeout
Time calculation by python method
Unknown Flaws
End

Figure 4.11 Flow Chart of Proposed Script

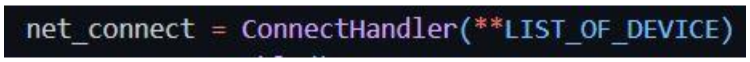
Figure 4.11 illustrates about the process of proposed script in this experiment , how script will run and manage configuration from start to end, there have lots of challenge in run time of application such “read IP” address , if IP addresses store in a YAML/JSON file or in a dictionary it must be assigned on exact router, after running the script in automation environment script will look for destinated IP address where this host is located to configure it, if destinated host IP is incorrect script will be fail to run , but there have a solution if script fail in one host , script will skip this one and jump to automate next one according to code sequence. Not only for exact IP address also need SSH authentication and same way follow to automate. Three error can normally show such as authentication fail, timeout and unknown error due to wrong destinated host or fault authentication.

There have also some risks of failure for device restriction such as RESTCONF and NETCONF. Some of device do not support this technology. These two-network automation is really updated with high technology. RESCONF support HTTP request, able to send configuration by HTTP request also easy to make URL of device by python, by using “POSTMAN” it is very flexible to get output of device history with configuration assigning URL. Although in this project firstly showing only CLI based automation, but RESCONF and RESTCONF will be shown in this experiment shortly.

Figure 4.11 allow for all type of automation only device is changed due to router software such as Cisco IOS-XE support RESTCONF. Where IOS do not support RESTCONF by HTTP request. Various company still using IOS version router, so they use CLI based automation but new and large company updating their device to make low cost for automation purpose and replacing old router by IOS-XE such as CSR1000v router, it supports HTTP request. Latest automation reduces human activity and time savings which is most significant for a company financially.

3.4.7.0 Connection Establish

There have one netmiko method which establish connection between code editor and device by calling SSH (Secure Shell)



```
net_connect = ConnectHandler(**LIST_OF_DEVICE)
```

The crucial method name is “ConnectHandler” and it is mandatory for netmiko[CLI BASED] to establish connection and check authentication.

3.5.1.0 RESTCONF

Using the HTTP protocol, A programmatic interface known as RESTCONF provides access to arrangement data, state-run data, RPC-Remote Procedure Call activities, and experiences specified in the YANG standard.

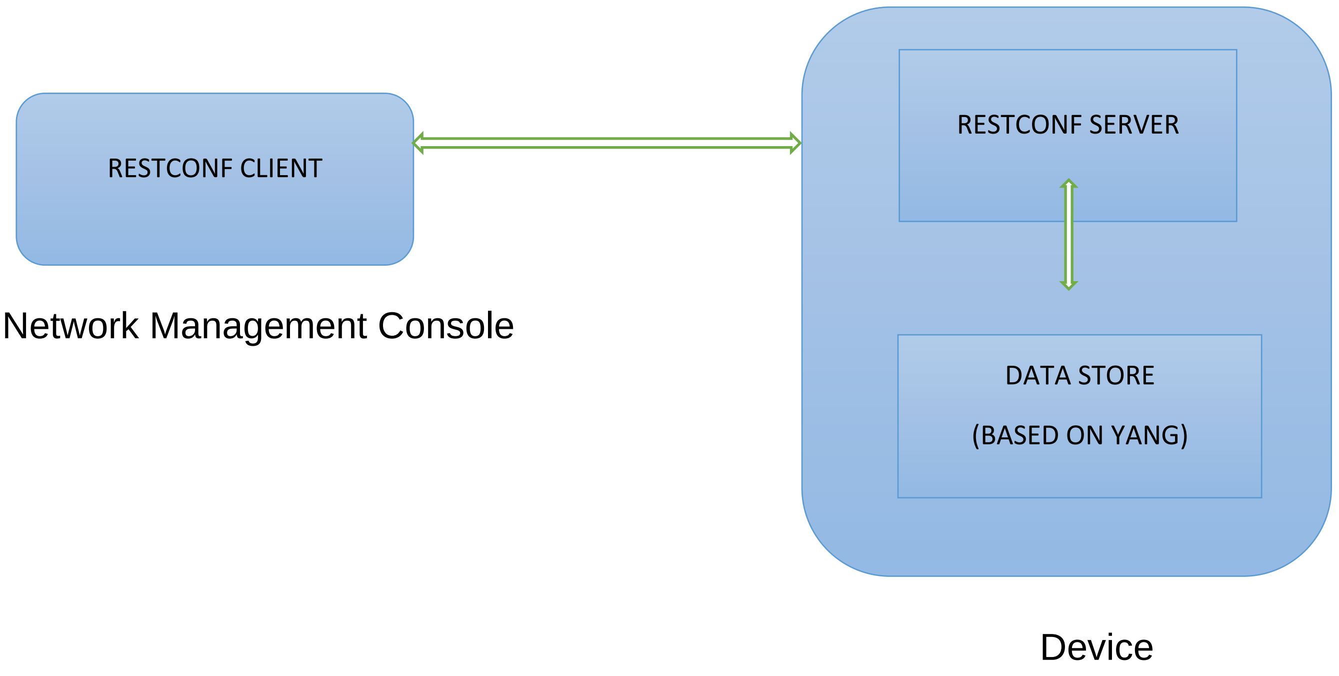


Figure 4.12 RESTCONF

Figure 4.12 shows process of RESTCONF automation and nowadays it is the most famous mechanism of network automation where it makes use of HTTP request-based REST APIs, the most widely utilized practice in the contemporary automation period.

REST API stands for representation state transfer, it is a development of WWW. RESTCONF operation used by HTTP there have various methods available such:

- GET
- POST
- PUT
- PATCH
- DELETE

3.5.1.1 Main Goal of RESTCONF Automation in this project

In this part it is examined that how automation possible in network devices using HTTP based requests utilized python programming language. Main goal of this automation is configuring router interfaces and DHCP Server by HTTP requests method using python programming. According to research questions this part also done. After sowing all procedure of RESTCONF network automation the result will be expressed in “Result” section. This automation part will demonstrate how RESTCONF Network automation is beneficial and meet with research questions with its all attribute and usage.

3.5.2.0 RESTCONF OPERATION

RESTCONF based on HTTP, therefore its operation executed by URLs. There have several sections available task by task.

- ✓ Address: The RESTCONF server's IP address is also for a network device.
- ✓ Root: Central access position of RESTCONF request. Considered as a root directory.
- ✓ Data: Resource data of RESTCONF API.
- ✓ YANG Module : The YANG data model and container, which specifies the commands that will be utilized, will be employed.
- ✓ Leaf: It isa separate segment, and it is found in container.
- ✓ Option: It is an optional parameter and fetch effect on result.

What is YANG data model?

YANG is a data modelling language, utilized to prototype state data, alignment and administrative executions manipulated by NETCONF protocol. YANG published in September 2020 as RFC 6020.In network automation YANG data model is very crucial only for IETF interface, IETF stands for Internet Engineering Taskforce, for automation purpose application usually connect with IETF interface to push data or get data, data come with 2 format such as XML or JSON.

3.5.3.0 NETWORK AUTOMATION USING RESTCONF

According to research question to demonstrate network automation, a primary network topology is created based on RESTCONF network automation which procedure is more exceptional than CLI based automation. This automation process is latest mechanism of real-world network technology.

3.5.4.0 Topology For RESTCONF

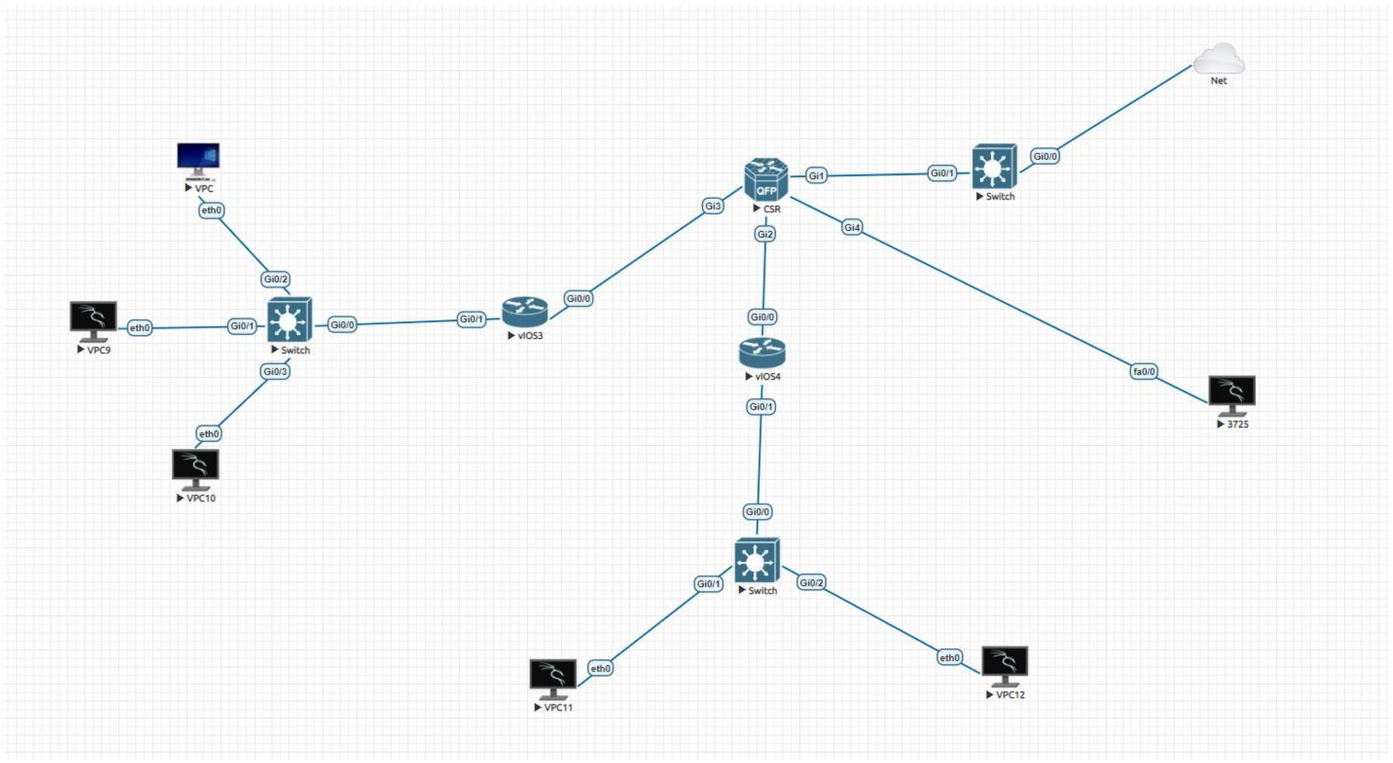


Figure 4.13 RESTCONF Topology

Above figure shows about the configuration of network automation using RESTCONF also known as API based network automation. This topology is used to demonstrate how flexible and effective RESTCONF Network automation than other.

3.5.5.0 Topology Overview for RESTCONF

RESTCONF network automation is the latest modern network automation mechanism only for HTTP request and its data model, but it support some certain of devices, maximum company migrating their infrastructure to those devices which support RESTCONF network Automation.

According to Cisco their have various kind of build-in software for their devices such as IOS, IOS-XE and so on. Only IOS-XE software support RESTCONF network automation and one of routers name is “CSR100v” which contain IOS-XE, and it is used in Figure 4.13. entitled as “CSR”.

The target router of this topology is “CSR100v”, is configured by python script and creating URL of router for HTTP requests. In earlier has only mentioned about RESTCONF methods, there have some details of each method, it is important to discuss about method because of this method implemented in python script for automation purpose:

Method Name	**GOAL**
Get	Utilize for Read
Patch	Use for Update configuration.
Put	Use for Create or Replace any update.
Post	Apply for Create or Operations.
Delete	To remove or delete any resources.

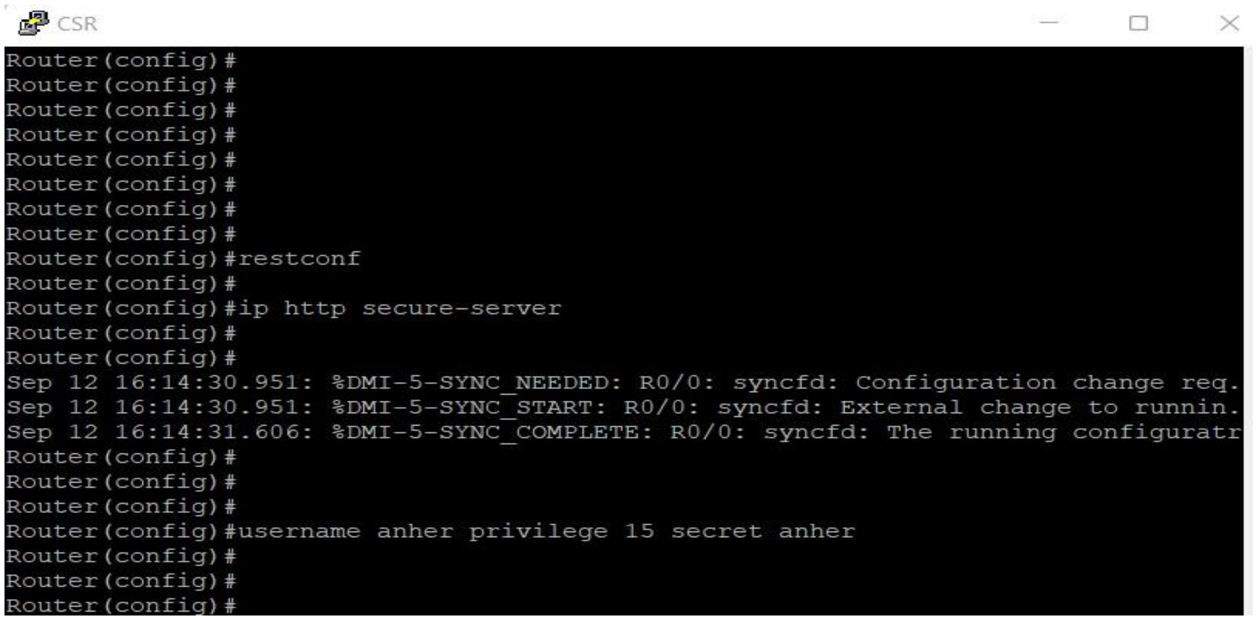
Table 4.4 RESTCONF Methods

3.5.6.0 Procedure of RESTCONF Network Automation

To implement RESTCONF in this experiment here used some potential command in “CSR 1000v” router to allow HTTP request and establish SSH connection for authentication. Following commands is used in CSR (Cloud Service Router) router:

- restconf [Enable API for accessing data in YANG, including IETF interface to communicate with application]
- ip http secure-server [Enable http requests from user interface]

- ➢ username anher privilege 15 secret anher [Enable SSH for security purpose]



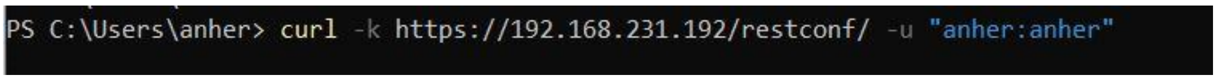
```
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#
Router(config)#restconf
Router(config)#
Router(config)#ip http secure-server
Router(config)#
Router(config)#
Sep 12 16:14:30.951: %DMI-5-SYNC_NEEDED: R0/0: syncfd: Configuration change req.
Sep 12 16:14:30.951: %DMI-5-SYNC_START: R0/0: syncfd: External change to runnin.
Sep 12 16:14:31.606: %DMI-5-SYNC_COMPLETE: R0/0: syncfd: The running configuratr
Router(config)#
Router(config)#
Router(config)#
Router(config)#username anher privilege 15 secret anher
Router(config)#
Router(config)#
Router(config)#
```

Figure 4.14 Initial Command for RESTCONF

Above 3 commands are given manually to CSR (Cloud Service Router) to make the way of RESTCONF Network Automation.

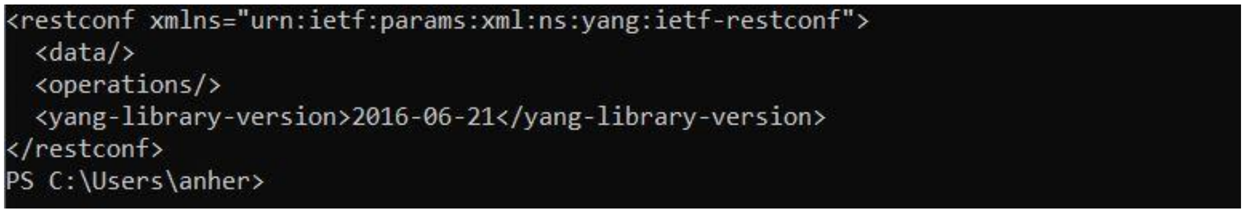
Management interface always significant for automation it works as destinated device. Management interface established by DHCP Server from management cloud. "Interface GigabitEthernet 1" is enabled as management interface and obtained IP address "192.168.231.192" from DHCP Server of Virtual Machine, also static configuration allows to make management interface. After creating management interface, it is mandatory to check whether does router providing https response from command line (power shell) or Linux Root by this following command including management interface:

- ➢ *Curl -k https//192.168.231.192/restconf/ -u "anher: anher"*



```
PS C:\Users\anher> curl -k https://192.168.231.192/restconf/ -u "anher:anher"
```

Response from router:

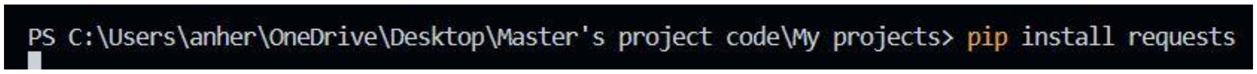


```
<restconf xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf">
  <data/>
  <operations/>
  <yang-library-version>2016-06-21</yang-library-version>
</restconf>
PS C:\Users\anher>
```

Response have come with this keyword “yang: ietf-restconf”, its means router providing response successfully of http request and allowing configuration by https request.

3.5.7.0 Proposed PYTHON Script of RESTCONF

After confirming http requests and above initial configuration it is time to write scripts. Here also python programming language used to automate CISCO IOS-XE – CSR (Cloud Service Router). This automation also done by remembering of research questions. For RESTCONF network automation here implemented one of famous python library known as “requests”. To install this library simply used one command “pip install requests” from Visual Studio Code Terminal.

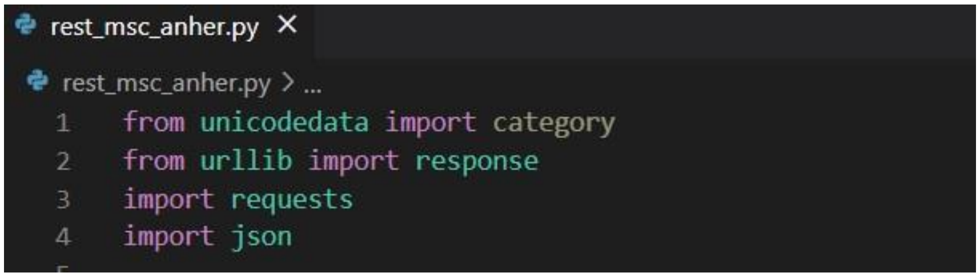


```
PS C:\Users\anher\OneDrive\Desktop\Master's project code\My projects> pip install requests
```

Requests library is mandatory for RESTCONF network automation to write and implement code.

Response can exhibit by two types of data format such as XML and JSON, in this experiment JSON (Java Script Object Notation) is utilized. Another segment is “URL” and it is one of best facility of RESTCONF. Illustrating URL by python of a router and use this URL to get update later, this mechanism is the key part of RESTCONF.

Above all consideration, 2 library and 1 data format are imported in python script.



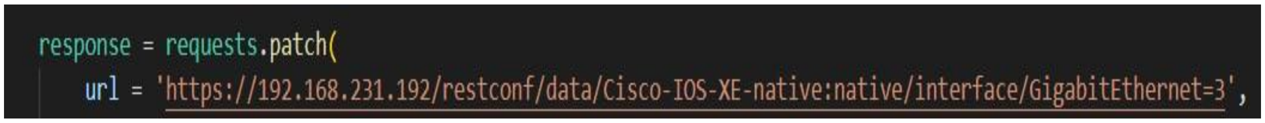
rest_msc_anher.py

rest_msc_anher.py > ...

```python
from unicodedata import category
from urllib import response
import requests
import json
```

After declaring about libraries and data format for output, it is significant to call the router via its "URL" .

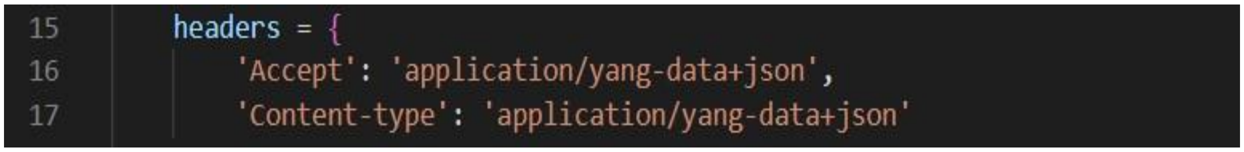
There have one question, how "URL" is generated of the router? "URL" is generated of a router by its Management IP address, and it's enabled by a command "ip http secure-server" earlier.



```python
response = requests.patch(
    url = 'https://192.168.231.192/restconf/data/Cisco-IOS-XE-native:native/interface/GigabitEthernet=3',
```

Above URL shows that the target router is "192.168.231.192" and configuring interface "GigabitEthernet 3" of this router.

There also have another significant part it is "header", how data will be responded or how data will show as output after using "URL" to router. This amazing feature is declared in script what type of data format will be respond as "header".



```python
    headers = {
        'Accept': 'application/yang-data+json',
        'Content-type': 'application/yang-data+json'
```

According to my script it is clearly expressed that data format is assigned as "JSON".

3.5.8.0 Data Sets

4 sets of data assigned in this experiment. For SSH authentication used my name as username and password. Some of IP address obtained dynamically from DHCP Server of Virtual Machine and some of IP address assigned statically.

Path	Address	Username	Password	Type
Management IP	192.168.231.192 /24	anher	anher	dynamic

CSR Router Interface G2	192.168.235.234 /24	anher	anher	static
CSR Router Interface G3	12.12.12.1 /24	anher	anher	Static to retrieve
DHCP SERVER	172.16.1.0 /24	anher	anher	Dynamic
DNS SERVER	9.9.9.9	anher	anher	static
DEFAULT ROUTER	172.16.1.1	anher	anher	static

Table 4.5 Data Sets of RESTCONF

Above Table 4.5 shows data set of RESTCONF network automation. Some data is assigned dynamically after automation such as DHCP Server setup and Interface configuration by RESTCONF python automation.

In this experiment I have created a URL of router https://192.168.231.192/restconf/data and assigned in my python script. This IP address is the management interface of the router (CSR1000v), I used in my RESTCONF lab. Therefore, using this URL I can check running configuration by a free software called **"POSTMAN"** using API request and support two data format such as JSON and XML for providing result**.** This software designed for Application Programming Interface (API) call, also user can use all methods of **requests library** such as GET, POST, PUT, PATCH, DELETE

4.0.0 Experimentation and Result

Two result is provided for two type of network automation specifically "CLI based Network Automation" and "RESTCONF Network Automation", all experiment demonstrated by keeping research questions in mind.

4.1.0 Result of CLI based Network Automation

Elapsed Time of Automation

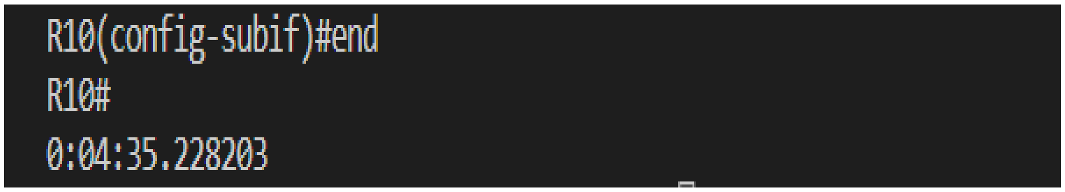
1. After automating routers by python script it shows an unexpected result which is really tremendous, 27.5 seconds take configuration time for each router.

2.Manual configuration takes 6 minutes per router, which is really ineffective.

3.Connection establishing time 4 seconds from automation tool to network device.

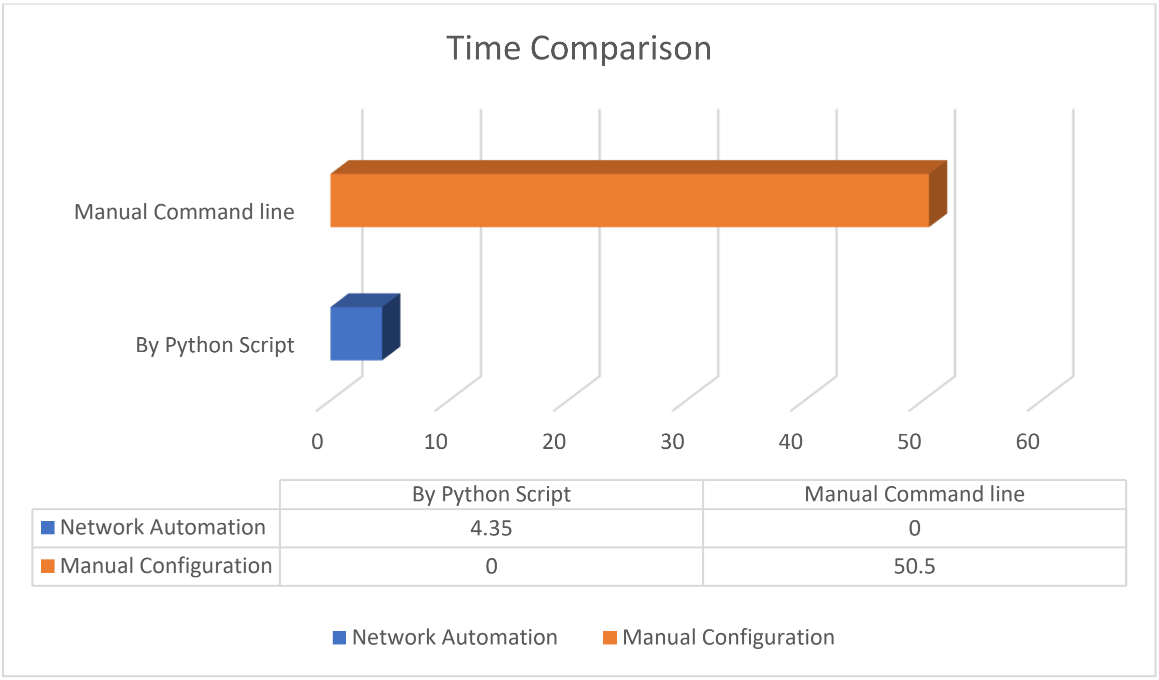
4.Banner setup time of all routers is 12 seconds with router description.

5.Total configuration time of 10 routers is 4.35 minutes or 275 seconds. These results show using time **timeit** and **timedelta** libraries in python.



```
R10(config-subif)#end
R10#
0:04:35.228203
```

Figure 5.1 Elapsed time of full automation.



	By Python Script	Manual Command line
🟦 Network Automation	4.35	0
🟧 Manual Configuration	0	50.5

Figure 5.2 Time Comparison of Network Configuration

Above Figure 5.2 illustrates about the comparison between Manual Network configuration and Network automation using python programming language. Manual configuration of network device is time consuming, network administrator needs to provide same command line many times in same device and other devices, where network automation using python programming can reduce time using dynamic methods.

An example from python network automation expressed in this project, Configuring routing protocol OSPF in 10 routers. Manual configurations need same command line in every router, and it is really time consuming and boring for network manager. But python did it in this project very simply by writing couple of lines in 10 routers.

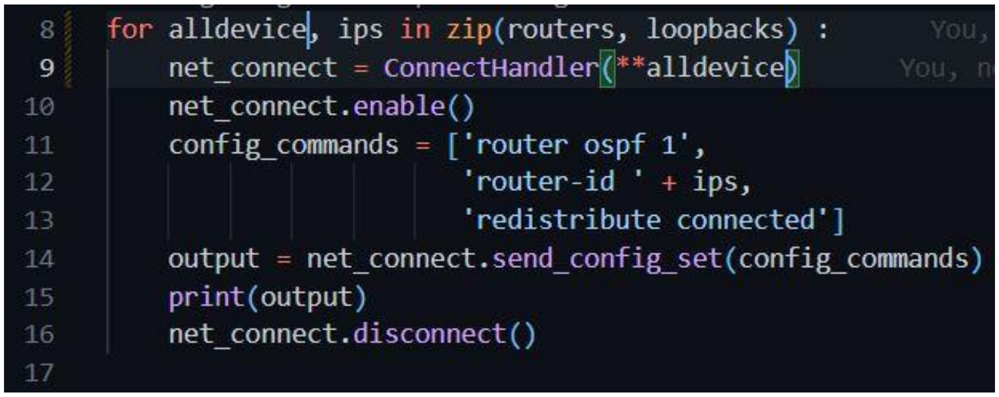

```
for alldevice, ips in zip(routers, loopbacks) :
    net_connect = ConnectHandler(**alldevice)
    net_connect.enable()
    config_commands = ['router ospf 1',
                       'router-id ' + ips,
                       'redistribute connected']
    output = net_connect.send_config_set(config_commands)
    print(output)
    net_connect.disconnect()
```

Figure 5.3 Time saving configuration by python in 10 routers

Above Figure 5.3 expressed about OSPF routing protocol configuration by python, only these few python lines can configure OSPF routing protocol in 10 routers in a few minutes, also no keyword has been repeated but in manual network configuration same command line or configuration repeated 10 times for 10 routers.

4.1.1 Data Minimization

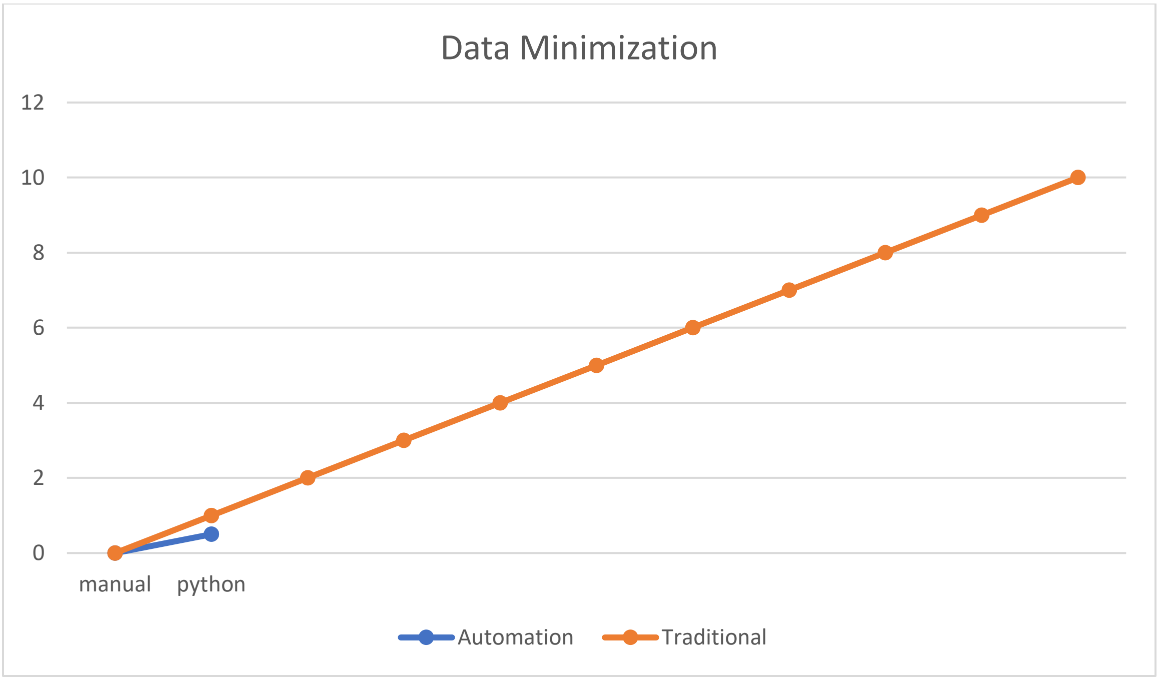
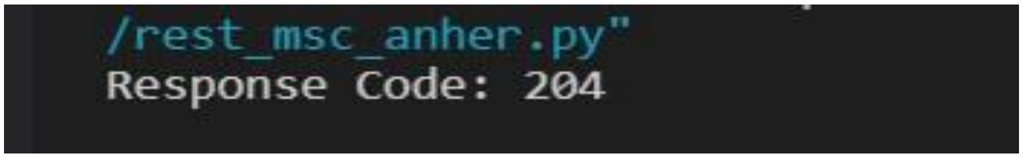


Figure 5.4 Data Minimization

Above Figure 5.4 interpret about the minimization of data, to configure router OSPF in 10 router **“router ospf 1”** have to put this command for each router in heterogenous network, in example 100 times need to write same command for 100 router by using manual configuration, but in network automation it is easy to provide a range using **for loop**, no matter how many routers or storing all routers in a dictionary and calling dictionary from main script using **for loop**.

4.2.0 Result of RESTCONF Network Automation

After HTTP based network automation, I obtained a response code “204”, basically in restconf network automation output come across two way one is “204/201/200” and another response code is “404”, if it shows “204” that is mean automation is successful and if code is “404”, the automation has not been successful.

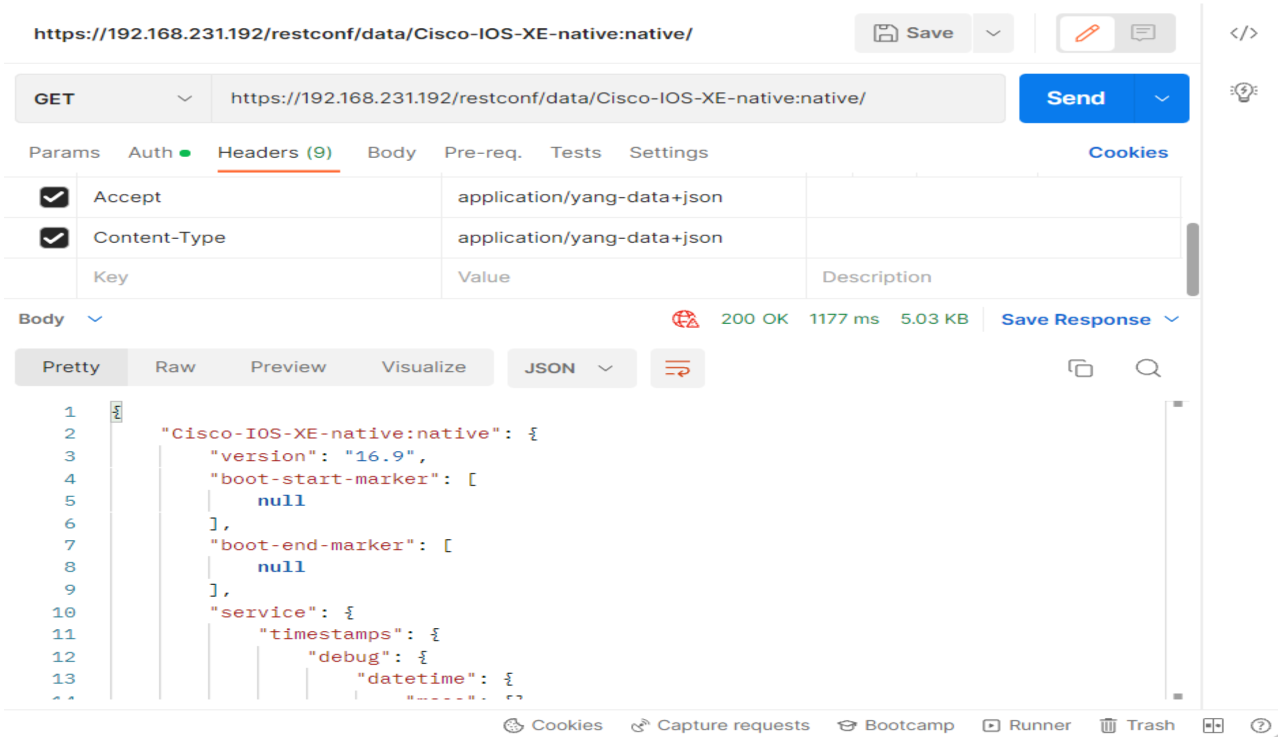


```
/rest_msc_anher.py"
Response Code: 204
```

Figure 5.5 Output of RESTCONF Network Automation.

4.3.0 Using postman to test URL and running configuration

HTTP based automation using router URL https://192.168.231.192/restconf/data takes only 1176 [Fig 5.7] millisecond to automate a CSR1000v router, which is a tremendous mechanism in tech world. Also, it is very flexible to monitor running configuration or checking performance by API call using POSTMAN (free API call software).



https://192.168.231.192/restconf/data/Cisco-IOS-XE-native:native/

GET https://192.168.231.192/restconf/data/Cisco-IOS-XE-native:native/

	Key	Value	Description
✓	Accept	application/yang-data+json	
✓	Content-Type	application/yang-data+json	

200 OK 1177 ms 5.03 KB

```
{
    "Cisco-IOS-XE-native:native": {
        "version": "16.9",
        "boot-start-marker": [
            null
        ],
        "boot-end-marker": [
            null
        ],
        "service": {
            "timestamps": {
                "debug": {
                    "datetime": {
```

Figure 5.7 Testing configuration by POSTMAN

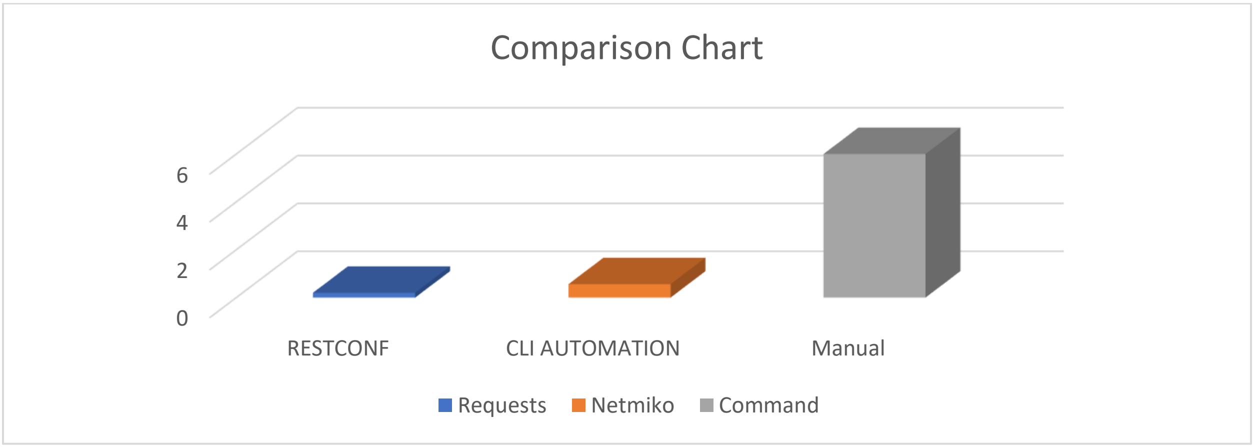


Figure 5.6 Comparison of Network Deployment

Figure 5.6 demonstrate about the comparison about overall performance of all mechanism of network deployment. RESTCONF take less time than other mechanism and exhibit the output of a router within 1177 millisecond by REST API call. CLI based network automation is little bit slow than RESTCONF but Industrially CLI automation is still being used for expense of RESTCONF supported device.

4.4.0 Comparison of Proposed Automations to other

Authors	Number of Device	RESTCONF	Time Elapsed in Second (Per Device)
[8]	1	No	12
[6]	**6**	No	8
[5]	**8**	NO	11.5
[3]	**1**	No	7
Proposed Automation	10	YES	5
Proposed Data	Dictionary	JSON	

Table 5.1 Comparison with other authors

Proposed Script faster than other research only for storing data in dictionaries. In addition, RESTCONF network automation also exhibited along with CLI based automation , where it is the magnificent mechanism in network automation by its speed.

5.0 Evaluation

According to research questions it is effectively demonstrated in this experiment that it is possible to deploy heterogenous network by one network automation engineer using python programming in spite of that after all research it is proved network automation has various criteria for automate devices by specific vendor and devices. In addition, for a large network infrastructure, company or IT manager need more NOC (Network Operation Engineer) engineer to configure all router manually using command line interface by manual command, which is really expensive for an organization to recruit more engineer for same tasks, According to **RQ1** it is proved that I configured 10 routers within few minutes which is significantly beneficial for industry to reduce the employee for same tasks, economically it is highly suggested. But this research exhibit that if one network automation engineer writes a script for network automation purpose using python programming it is possible to configure multiple devices in a short time with providing high security including SSH authentication. This research exemplifies that “network automation can minimize human involvement in heterogenous network”.

According to **RQ3**, it is proved that proposed investigation takes only 4 minutes to configure all devices and it is the most significant value for a company to save times.

Also, it is expressed that two most significant value of network automation are “Time Savings” and “Low Cost”. In this project CLI based network automation and RESTCONF both categories illustrate it, where CLI based network automation takes only 4.35 minutes and RESTCONF automation done by HTTP requests takes only few second which is really time savings, on the other hand only one engineer can deploy the script and can configure entire infrastructure which is effectively “low cost” for a company or organization.

I used python programming language because of python has some certain libraries which are designed for network devices and establishing SSH connection between automation server and devices, where other language unable to do it such as PHP, JAVA SCRIPT AND JAVA, this statement meet with **RQ4**, Where all automation script successfully done by python programming language, and it is highly recommended only for its specific libraries for SSH connectivity. In same way remote configuration also applied by using **SSH Remote Plugin** in Visual Studio Code, after identifying target device by python Dictionary it is easy to troubleshoot or configure device by Remote SSH Extensions using **Visual Studio Code**, This statement meets with **RQ2**.

By comparing with both automation process, RESTCONF is the latest mechanism of network automation where it uses HTTP based automation using REST API also it takes only few seconds to configure, and it is exhibited in this research, troubleshooting mechanism also breathtaking where I used URL of router by POSTMAN, This result meet with **RQ5**, it demonstrates the best way of automation. Due to few versions of device support this automation, therefore CLI based network automation process is still common but maximum company updating their infrastructure and migrating to Cloud Service Router for RESTCONF mechanism. According to my research I prefer both automation process, if router support RESTCONF I would recommend RESTCONF if router do not support, I would recommend CLI based automation according to my research.

Above all statements are fully aligned with the research questions **RQ1, RQ2,RQ3,RQ4** and **RQ5**, the outcome of my experiment regarding network automation using python programming.

6.0 Conclusion

Network infrastructure is a crucial part in every company and organization, nowadays maximum data centers are hybrid including cloud and own infrastructure for high reliability network. Concurrently, this is really expensive for a company where company also need network engineer to maintain infrastructure. If company hire more engineer which is not cost worthfully, in this instance Network Automation Engineer is really crucial for an organization or a company, where one engineer able to do all configuration and monitoring by using python programming in example, only one Network automation engineer can configure all of devices using CLI based network automation according to my research.

Exceptionally done in my proposed script: According to my research I stored 10 routers in python dictionary, therefore network automation engineer able to store unlimited devices including SSH authentication using python dictionary {curly braces}, which is really beneficial for automation purpose for an organization.

.

My proposed script and automation procedure illustrate how network automation is really beneficial for a company and how it minimizes people for same task, no need extra engineer for same tasks, python code is reusable. It is potentially possible to reuse same code for another router or another environment just changing IP address and SSH authentication.

Network monitoring also a crucial part of every industry in daily basis. Performance Checking, Troubleshooting, Device Updating, Device Backup, Data Load and so on. Also, all of these tasks take more time which is not time savings for a corporate large company. Network automation ease this process and provide outcome in a quick response such as using RESTCONF, and it provides URL of devices. To get the performance or running configuration of a device, URL is the stunning mechanism to get quick response in a few seconds, which procedure effectively beneficial for an organization to save times. In my project it exemplified that how showing running configuration of a device using URL in POSTMAN by REST API call. These troubleshooting processes are the present of network automation. Therefore, maximum corporate company migrating their infrastructure to Cloud Service Devices such as CSR1000v Router whereby it supports RESTCONF network automation.

Finally, update technology constantly beneficial for all organization to minimize cost, reduce maintenance time and so on, whereby lots of vendor updating their products to support latest mechanism, which is more aligned with modern technological world, software developing technology is being more advance and to align with those technology network automation is a big example in IT sector and facility potentially uncountable.

6.1 Future Work

Accomplishing all research this project successfully finished based on network automation where from long time network engineer accustomed to do configuration by command line interface which is really ineffective in counting various way. However, after finishing all research I realized it is possible to establish more extra resources for network automation and this research will be on going, there have some specific plans:

1. Developing an IOT device which will be able to do device configuration by voice command.
2. Developing a software for all network devices which will be used by USB flash drive in network devices to create URL for network monitoring and management from anywhere by any end device.
3. Establishing two factor authentication in network devices along with SSH.

Although above plan will take more time, but I will finish step by step with providing my best effort to provide an innovative outcome in network automation sector.

Thank you very much Honorable Professor for your support, time and consideration.

Bibliography

[1] A. -. CISCO, https://www.cisco.com/c/dam/en/us/products/collateral/cloud-systems-management/network-services-orchestrator/acg-economic-benefits-of-network-automation.pdf.

[2] Dyer, Matthias, Thomas Kalt and Jan Beutel, "Deployment support network." European Conference on Wireless Sensor Networks., Springer, Berlin, Heidelberg, 2007., pp. pp. 195-211.

[3] A. A. R. R. a. K. M. Mazin, Performance Analysis on Network Automation Interaction with Network Devices Using Python, In 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics, 2021.

[4] B. O. U. M. E. Z. R. A. G. Abdelhak, "A Proposed of Novel Network Management Platform for Network Automation and Programmability with Implementation on GNS3.", University of Mohamed Khider Biskra, 2020., 2020.

[5] J. Puumala, Automation of Router Configuration., IEEE, 2017.

[6] D. a. L. P. Gedia, "A Centralized Network Management Application for Academia and Small Business Networks.", Information Technology in Industry 6, no. 3 (2018), 2018.

[7] B. A. A. M. M. X.-N. N. K. O. a. T. T. Nunes, "A survey of software-defined networking: Past, present, and future of programmable networks." IEEE Communications surveys & tutorials 16, no. 3 : 1617-1634., IEEE, 2014.

[8] J. Larsson, Network Automation in a Multi-vendor Environment., 2020.

[9] S. RS, Automation in 5G. In5G Mobile Core Network pp. 277-300, Apress, Berkeley, CA., 2021.

[10] A. L. V. M.-M. J. P. M. L. D. a. M. V. 2. Aguado, Virtual network function deployment and service automation to provide end-to-end quantum encryption. Journal of Optical Communications and Networking, 10(4), pp.421-430..

[11] M. A. Z. K. M. &. S. S. I. Arifin, Automation security system with laser lights alarm on web pages and mobile apps, In 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE) (pp. 287-292). IEEE, 2019.

[12] M. D. K. C. F. R. G. S. S. P. O. A. F. H. d. M. M. W. H. D. S. a. J. J. Ehrlich, "Software-defined networking as an enabler for future industrial network management.", IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), vol. 1, pp. 1109-1112. IEEE, 2018.

[13] F. A. R. A. R. M. K. a. A. I. Daud, "Performance of encryption techniques using dynamic virtual protocol network technology.", IEEE 8th International Conference on System Engineering and Technology (ICSET), pp. 29-34, 2018.

[14] J. a. T. M. Antony, "Analysis of Ethernet Control Network.", IETE Journal of Research (2021): 1-9..

[15] M. F. Sanner, "Python: a programming language for software integration and development.", J Mol Graph Model 17.1 (1999): 57-61..

[16] S. a. B. H. Hörning, "RMWSPy (v 1.1): A Python code for spatial simulation and inversion for environmental applications.", Environmental Modelling & Software 138 (2021): 104970..

[17] P. B. T. C. R. &. S. F. Mihăilă, Network Automation and Abstraction using Python Programming Methods, MACRo 2015, 2(1), 95-103..

[18] M. A. Y. Yahya, SDN improvements and solutions for traditional networks., MS thesis. Çankaya Üniversitesi, 2017.

[19] R. a. R. N. 2. Wiryawan, Pengembangan aplikasi otomatisasi administrasi jaringan berbasis website menggunakan bahasa pemrograman python, Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer, 10(2), pp.741-752.

[20] A. R. A. F. A. a. M. K. Rochim, As-RaD System as a Design Model of the Network Automation Configuration System Based on the REST-API and Django Framework, Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, pp.291-298..

[21] M. F. P. M. a. M. L. Islami, "Implementation of Network Automation using Ansible to Configure Routing Protocol in Cisco and Mikrotik Router with Raspberry PI.", Jurnal Ilmiah KOMPUTASI 19, no. 2 (2020): 127-134..

[22] B. O. U. M. E. Z. R. A. G. Abdelhak, "A Proposed of Novel Network Management Platform for Network Automation and Programmability with Implementation on GNS3.".

[23] B. Choi, "Introduction to Python Network Automation." Introduction to Python Network Automation., Apress, Berkeley, CA,. 1-21., 2021.

[24] D. a. P. L. Gedia, A Centralized Network Management Application for Academia and Small Business Networks., Information Technology in Industry, 6(3)., 2018.

[25] W. P. Hank, REST APIs Part 1: HTTP is for more than Web Browsing A Network, [Performance]. Cisco DevNet, 2017.

[26] R. a. G. J. Tischer, Programming and Automating Cisco Networks: A guide to network programmability and automation in the data center, campus, and WAN., Cisco Press, 2016.

[27] W. P. Hank, APIs are Everywhere...but what are they?, A Network Programmability Basics, [Performance]. Cisco Devnet, 2017.

[28] B. Aly, Hands-On Enterprise Automation with Python, UK, Packt Publishing Ltd, 2018.

[29] R. Abhishek, Practical Network Automation Second Edition, Uk: Packt Publishing, Uk: Packt Publishing Ltd., 2017.

[30] C. Eric, Mastering Python Networking, Third Edition, UK: Packt Publishing Ltd, 2020.

[31] R. Abhishek, Practical Network Automation Leverage the power of Python and Ansible to optimize, UK: Packt Publishing Ltd., 2017..

[32] W. D. Gerald, Network Resource Modules, [Performance]. Red Hat, Inc. .

Appendix:

1.0 Code for CLI based network automation

1.1Deployment:

```
#!/usr/bin/env python3
import sys
sys.path.append('./scripts')
import global_unit
import int_seg
import ospf
from datetime import datetime
time_commence = datetime.now()

exec('global')
exec('int_seg')
exec('ospf')

time_end = datetime.now()
print('Duration: {}'.format(time_end - time_commence))
```

1.2 Global Config

```
#!/usr/bin/env python3
from imaplib import Commands
import sys
from unittest import result

from ospf import LIST_OF_DEVICE
sys.path.append('./inventory')
from netmiko import ConnectHandler
from router import routers, host, loopbacks, tunnel_rtr

#This configuration for all routers
for LIST_OF_DEVICE, names in zip(routers, host) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['hostname ' + names,
                       'banner login ^',
                       'This Configuration created by Tahtihal Anher!',
                       'Msc Projec 19062288!!',
                       '^']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()
```

```
for LIST_OF_DEVICE, ips in zip(routers, loopbacks) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['int loop0',
                 'ip address ' + ips + ' 255.255.255.0',
                 'descr Created by Tahtihal Anher']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()


# standard Configurations among the tunnel routers.
for LIST_OF_DEVICE in tunnel_rtr :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['crypto isakmp policy 12',
                       'encr aes',
                       'authentication pre-share',
                       'group 6',
                       'hash md6',
                       'crypto isakmp key cisco address 0.0.0.0',
                       'crypto ipsec transform-set ESP_AES_SHA esp-aes esp-
sha-hmac',
                       'crypto ipsec profile TAHTIHAL_PROFILE',
                       'set transform-set ESP_AES_SHA']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()
```

1.3 OSPF Configuration

```
#!/usr/bin/env python3
from imaplib import Commands
import sys
from unittest import result
sys.path.append('./inventory')
from netmiko import ConnectHandler
from router import *

#Global OSPF
for LIST_OF_DEVICE, ips in zip(routers, loopbacks) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['router ospf 1',
                        'router-id ' + ips,
                        'redistribute connected']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Configuring all tunnel routers
for LIST_OF_DEVICE in tunnel_rtr :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface Tunnel0',
                        'ip ospf 1 area 0',
                        'ip ospf network point-to-multipoint']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#assigning IP addresses oto segment 146
for LIST_OF_DEVICE in Segment146 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.146',
                        'ip ospf 1 area 1']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

# segment 079
for LIST_OF_DEVICE in Segment79 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.79',
                        'ip ospf 1 area 2']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Segment 37
for LIST_OF_DEVICE in Segment37 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.37',
                        'ip ospf 1 area 2']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Segment 67
for LIST_OF_DEVICE in Segment67 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.67',
                        'ip ospf 1 area 2']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Segment 13
for LIST_OF_DEVICE in Segment13 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.13',
                       'ip ospf 1 area 4']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Segment 23
for LIST_OF_DEVICE in Segment23 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.23',
                       'ip ospf 1 area 5']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Segment 45
for LIST_OF_DEVICE in Segment45 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.45',
                       'ip ospf 1 area 0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Segment 58
for LIST_OF_DEVICE in Segment58 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.58',
                       'ip ospf 1 area 3']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#Segment 108
for LIST_OF_DEVICE in Segment108 :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.108',
                       'ip ospf 1 area 3']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()
```

```
#Config R9 Seg9 for Area 2
net_connect = ConnectHandler(**R9)
net_connect.enable()
Commands = ['interface gigabitethernet0/0.9',
                   'ip ospf 1 area 2']
result = net_connect.send_config_set(Commands)
print(result)
net_connect.disconnect()

#Seg 7 for Area 2
net_connect = ConnectHandler(**R7)
net_connect.enable()
Commands = ['interface gigabitethernet0/0.7',
                   'ip ospf 1 area 2']
result = net_connect.send_config_set(Commands)
print(result)
net_connect.disconnect()

#Area 3
net_connect = ConnectHandler(**R5)
net_connect.enable()
Commands = ['interface gigabitethernet0/0.5',
                   'ip ospf 1 area 3']
result = net_connect.send_config_set(Commands)
print(result)
net_connect.disconnect()

#Segment 8 for Area 3
net_connect = ConnectHandler(**R8)
net_connect.enable()
Commands = ['interface gigabitethernet0/0.8',
                   'ip ospf 1 area 3']
result = net_connect.send_config_set(Commands)
print(result)
net_connect.disconnect()

#Segment10 for Area 3
net_connect = ConnectHandler(**R10)
net_connect.enable()
Commands = ['interface gigabitethernet0/0.10',
                   'ip ospf 1 area 3']
result = net_connect.send_config_set(Commands)
print(result)
net_connect.disconnect()
```

1.4 Network Segments

```
#!/usr/bin/env python3
from imaplib import Commands
import sys
from unittest import result

from ospf import LIST_OF_DEVICE
sys.path.append('./inventory')
from netmiko import ConnectHandler
from router import *

#single_segment_routers for singlesegments
for LIST_OF_DEVICE, octets, lastoct in zip(singleSegment, singleSegmentoct,
singleSegmentlastoct) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.' + lastoct,
                       'encap dot ' + lastoct,
                       'ip add 112.1.' + octets + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()


#configuring segment 146
for LIST_OF_DEVICE, lastoctet in zip(Segment146, octet146) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.146',
                       'encap dot 146',
                       'ip address 112.1.146.' + lastoctet + '
255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#configuring  segment 79
for LIST_OF_DEVICE, lastoctet in zip(Segment79, octet79) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.79',
                       'encap dot 79',
                       'ip address 112.1.79.' + lastoctet + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()


for LIST_OF_DEVICE, lastoctet in zip(Segment37, octet37) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.37',
                       'encap dot 37',
                       'ip address 112.1.37.' + lastoctet + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#segment 67
for LIST_OF_DEVICE, lastoctet in zip(Segment67, octet67) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.67',
                       'encap dot 67',
                       'ip address 112.1.67.' + lastoctet + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#segment 100
for LIST_OF_DEVICE, lastoctet in zip(tunnel_rtr, tunneloct) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.100',
                       'encap dot 100',
                       'ip address 169.254.100.' + lastoctet + '
255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#segment 13
for LIST_OF_DEVICE, lastoctet in zip(Segment13, octet13) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.13',
                       'encap dot 13',
                       'ip address 112.1.13.' + lastoctet + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#segment 23
for LIST_OF_DEVICE, lastoctet in zip(Segment23, octet23) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.23',
                       'encap dot 23',
                       'ip address 112.1.23.' + lastoctet + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#segment 45
for LIST_OF_DEVICE, lastoctet in zip(Segment45, octet45) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.45',
                   'encap dot 45',
                   'ip address 112.1.45.' + lastoctet + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#segment 58
for LIST_OF_DEVICE, lastoctet in zip(Segment58, octet58) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.58',
                   'encap dot 58',
                   'ip address 112.1.58.' + lastoctet + ' 255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()

#segment 108
for LIST_OF_DEVICE, lastoctet in zip(Segment108, octet108) :
    net_connect = ConnectHandler(**LIST_OF_DEVICE)
    net_connect.enable()
    Commands = ['interface gigabitethernet0/0.108',
                   'encap dot 108',
                   'ip address 112.1.108.' + lastoctet + '
255.255.255.0']
    result = net_connect.send_config_set(Commands)
    print(result)
    net_connect.disconnect()
```

1.5 Dictionaries of Data Sets

```
R1 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.209',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R2 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.226',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R3 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.202',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R4 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.214',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R5 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.211',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R6 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.199',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R7 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.207',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R8 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.201',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

R9 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.205',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}
```

```
R10 = {
    'device_type': 'cisco_ios',
    'host': '192.168.231.204',
    'username': 'anher',
    'password': '19062288',
    'secret': 'secret1'
}

routers = [R1, R2, R3, R4, R5, R6, R7, R8, R9, R10]
tunnel_rtr = [R1, R2, R3, R4, R5]
tunneloct = ['1', '2', '3', '4', '5']
tunnelspokes = [R1, R2, R3, R4]
tunnelspokeoct = ['1', '2', '3', '4']
loopbacks_interfaces = ['1.1.1.1', '2.2.2.2', '3.3.3.3', '4.4.4.4',
'5.5.5.5', '6.6.6.6', '7.7.7.7', '8.8.8.8', '9.9.9.9', '10.10.10.10']
hosts = ['R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8', 'R9', 'R10']

singleSegment = [R5, R7, R8, R9, R10]
singleSegmentoct = ['5.5', '7.7', '8.8', '9.9', '10.10']
singleSegmentlastoct = ['5', '7', '8', '9', '10']

Segment146 = [R1, R4, R6]
octet146 =['1', '4', '6']

Segment79 = [R7, R9]
octet79 =['7', '9']

Segment67 = [R6, R7]
octet67 =['6', '7']

Segment37 = [R3, R7]
octet37 =['3', '7']

Segment13 = [R1, R3]
octet13 =['1', '3']

Segment23 = [R2, R3]
octet23 =['2', '3']

Segment45 = [R4, R5]
octet45 =['4', '5']

Segment58 = [R5, R8]
octet58 =['5', '8']

Segment108 = [R8, R10]
octet108 =['8', '10']
```

2.0 RESTCONF CODE

2.1 Interface Configuration by Rest API

```
from unicodedata import category
from urllib import response
import requests
import json

from urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(category=InsecureRequestWarning)

def printoutputAsJSON(bytes):
    print(json.dumps(json.loads(bytes), indent=2))

response = requests.patch(
    url = 'https://192.168.231.192/restconf/data/Cisco-IOS-XE-
native:native/interface/GigabitEthernet=3',
    auth = ('anher', 'anher'),
    headers = {
        'Accept': 'application/yang-data+json',
        'Content-type': 'application/yang-data+json'

    },
    data = json.dumps({
        'Cisco-IOS-XE-native:GigabitEthernet': {
            'ip':{
                'address':{
                    'primary':{
                        'address': '12.12.12.1',
                        'mask': '255.255.255.0'
                    }
                }




            }
        }

    }),
verify= False)

print('Response Code: ' + str(response.status_code))
print(response.text)
```

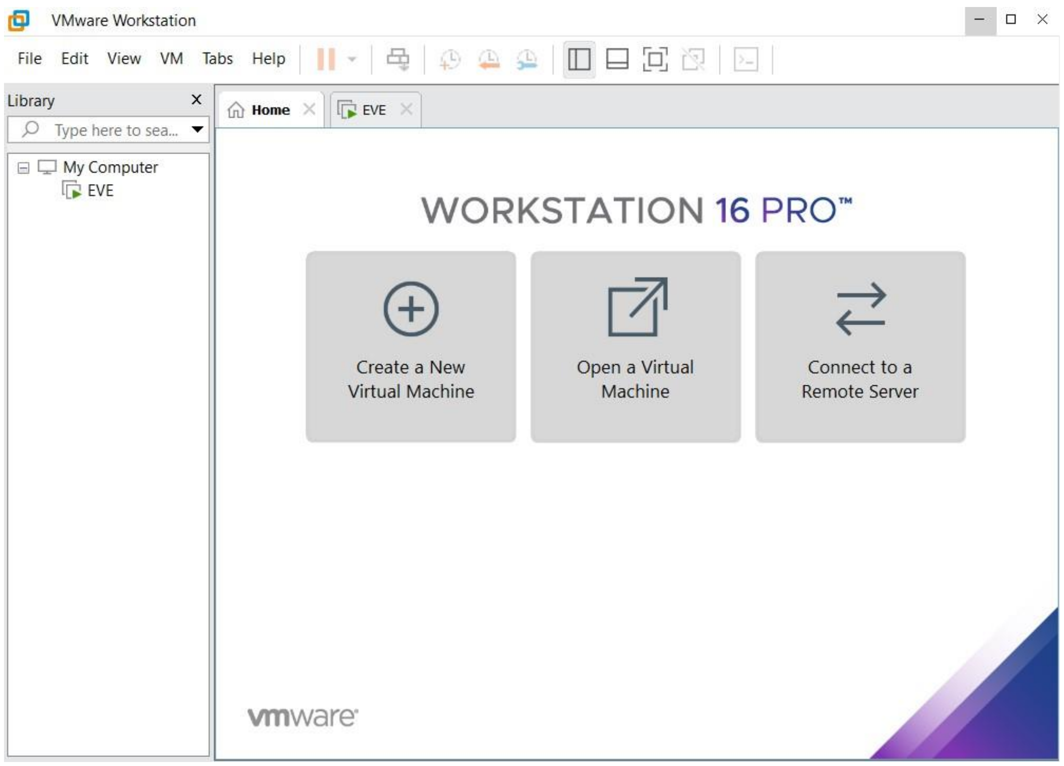
2.2 DHCP Configuration by Rest API

```
from unicodedata import category
from urllib import response
import requests
import json
```

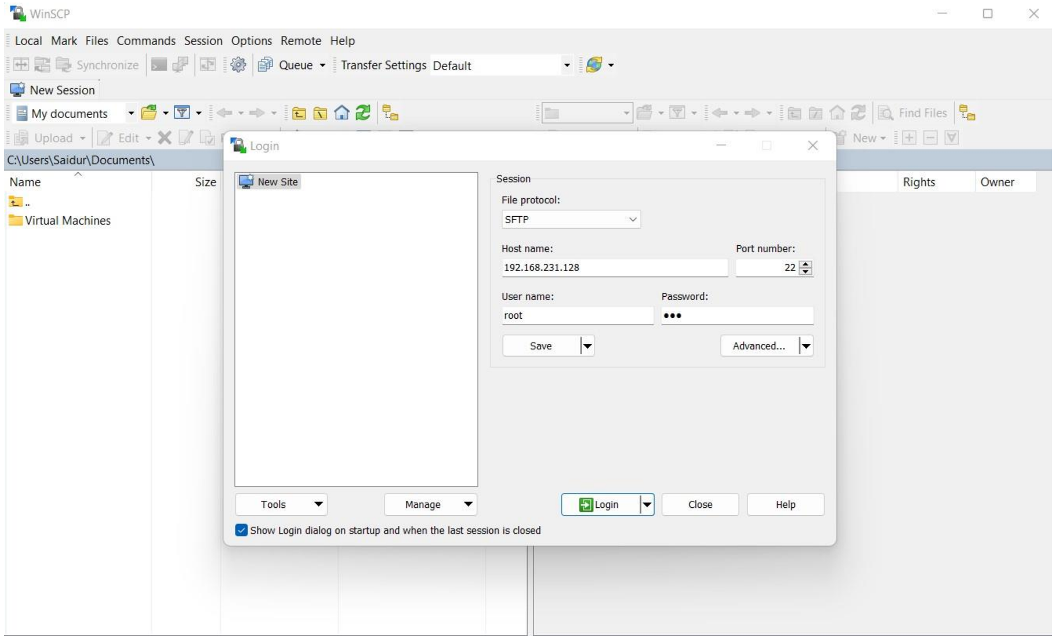
```python
from urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(category=InsecureRequestWarning)

def printoutputAsJSON(bytes):
    print(json.dumps(json.loads(bytes), indent=2))

response = requests.post(
    url = 'https://192.168.231.192/restconf/data/Cisco-IOS-XE-
native:native/dhcp:dhcp',
    auth = ('anher', 'anher'),
    headers = {
        'Accept': 'application/yang-data+json',
        'Content-type': 'application/yang-data+json'

    },
    data = json.dumps({
        "Cisco-IOS-XE-native:native": {

                "dhcp": {
                    "Cisco-IOS-XE-dhcp:excluded-address": {
                        "low-high-address-list": [
                            {
                                "low-address": "172.16.1.10",
                                "high-address": "172.16.1.20"
                            }
                        ]
                    },
                    "Cisco-IOS-XE-dhcp:pool": [
                        {
                            "id": "anher19062288",
                            "default-router": {
                                "default-router-list": [
                                    "172.16.1.1"
                                ]
                            },
                            "dns-server": {
                                "dns-server-list": [
                                    "9.9.9.9"
                                ]
                            },
                            "network": {
                                "number": "172.16.1.0",
                                "mask": "255.255.255.0"
                            }
                        }
                    ]
                },

    }
}),
verify= False)

print('Response Code: ' + str(response.status_code))
print(response.text)
```

3.0 Output and Screenshots of Whole Process

3.1 Virtualization



3.2 Images Importing to EVE-NG Virtual Machine



C:\Users\Saidur\Documents\

Session

File protocol: SFTP

Host name: 192.168.231.128

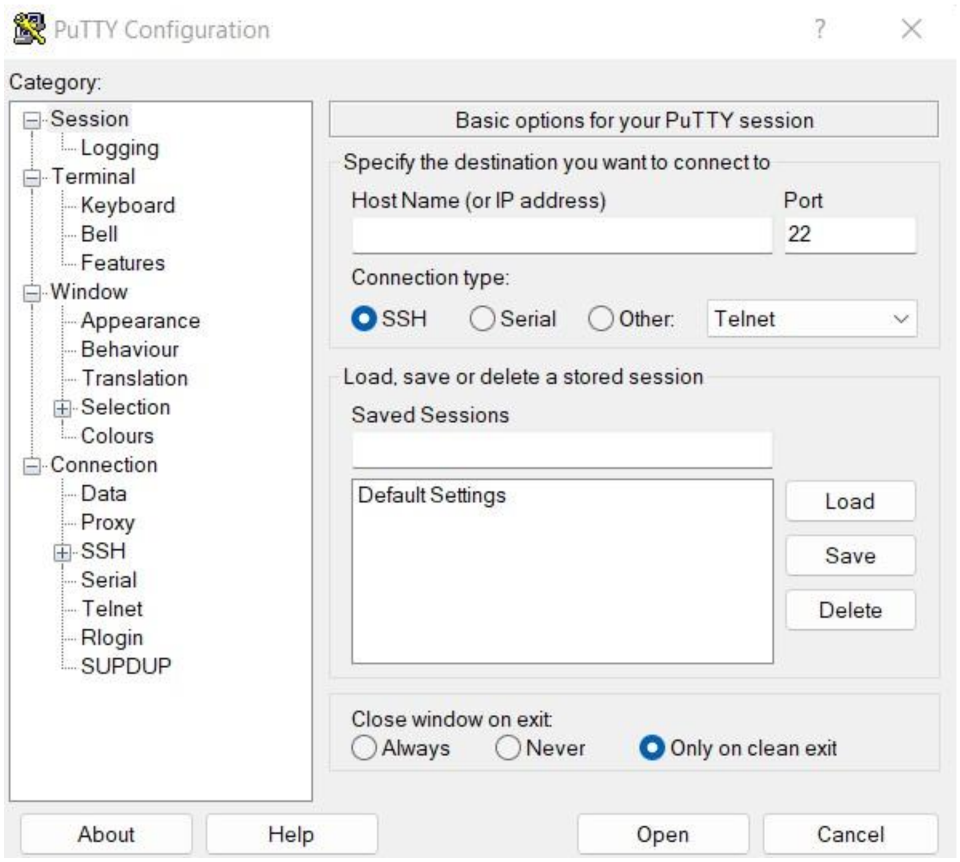
Port number: 22

User name: root

Password: •••

[✓] Show Login dialog on startup and when the last session is closed

3.3 PUTTY for SSH and Console



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

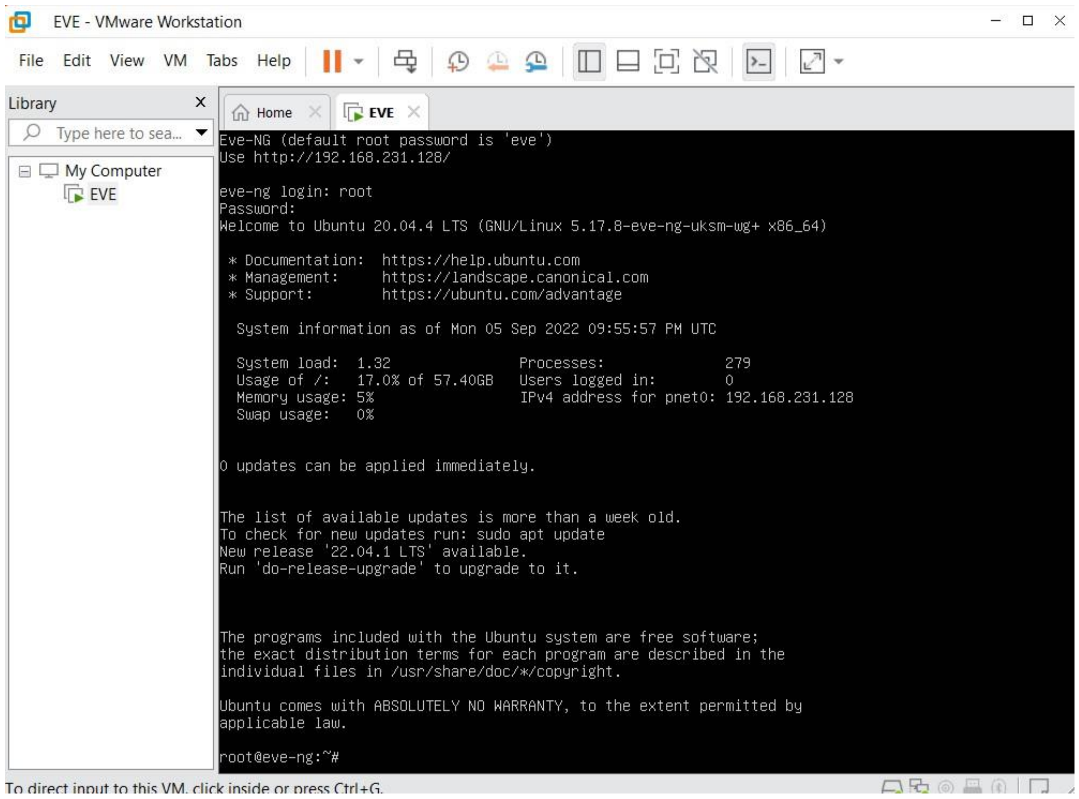
Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

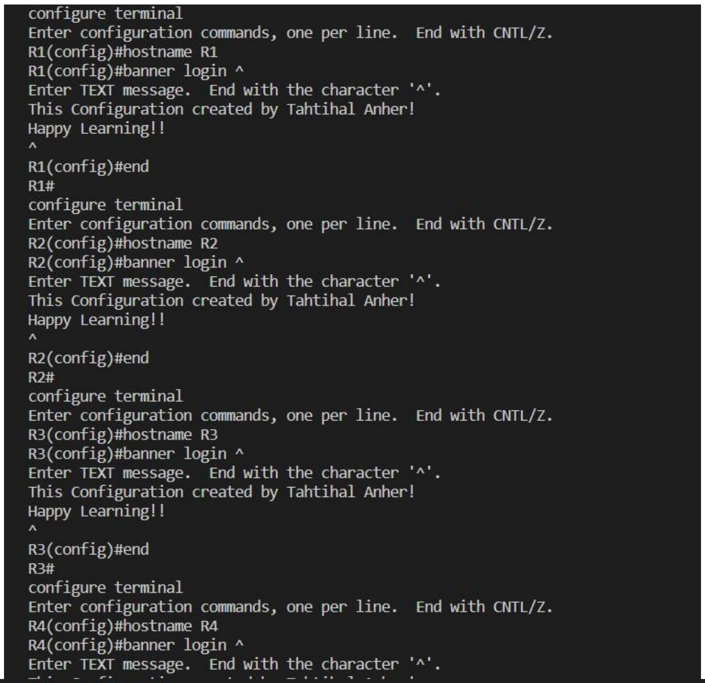
3.4 EVE-NG Screenshot (Run time in VM)


EVE - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to sea...
My Computer
EVE
Home
EVE
To direct input to this VM, click inside or press Ctrl+G.

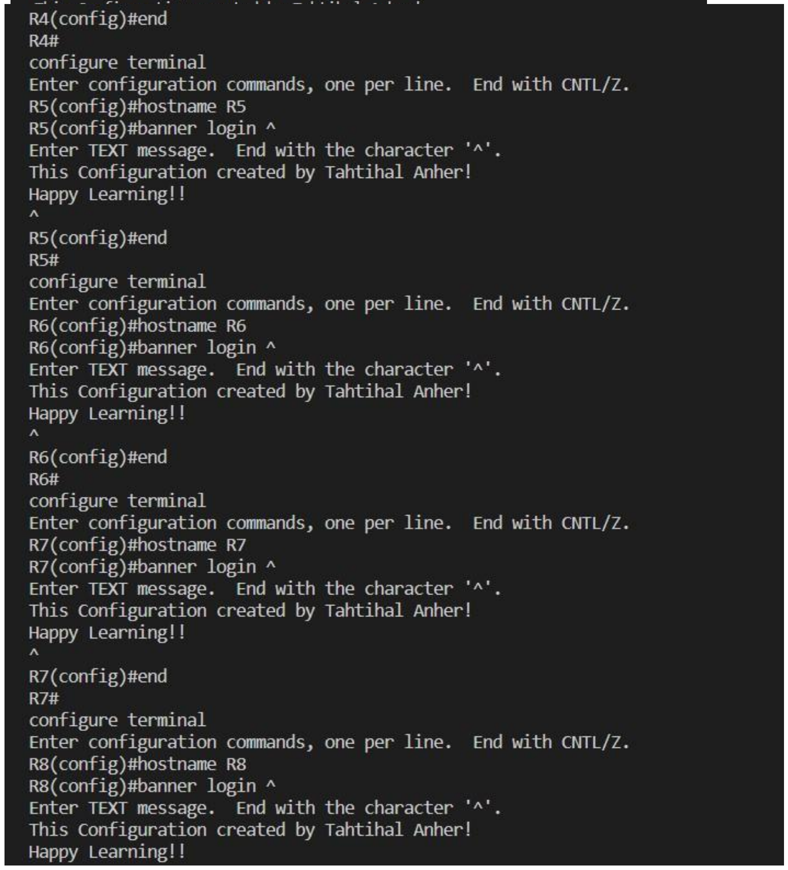
```
Eve-NG (default root password is 'eve')
Use http://192.168.231.128/

eve-ng login: root
Password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.17.8-eve-ng-uksm-wg+ x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon 05 Sep 2022 09:55:57 PM UTC

  System load:  1.32               Processes:              279
  Usage of /:   17.0% of 57.40GB   Users logged in:        0
  Memory usage: 5%                 IPv4 address for pnet0: 192.168.231.128
  Swap usage:   0%


0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@eve-ng:~#
```

3.5 Outputs of CLI BASED NETWORK AUTOMATION

Basic Descriptions of all routers

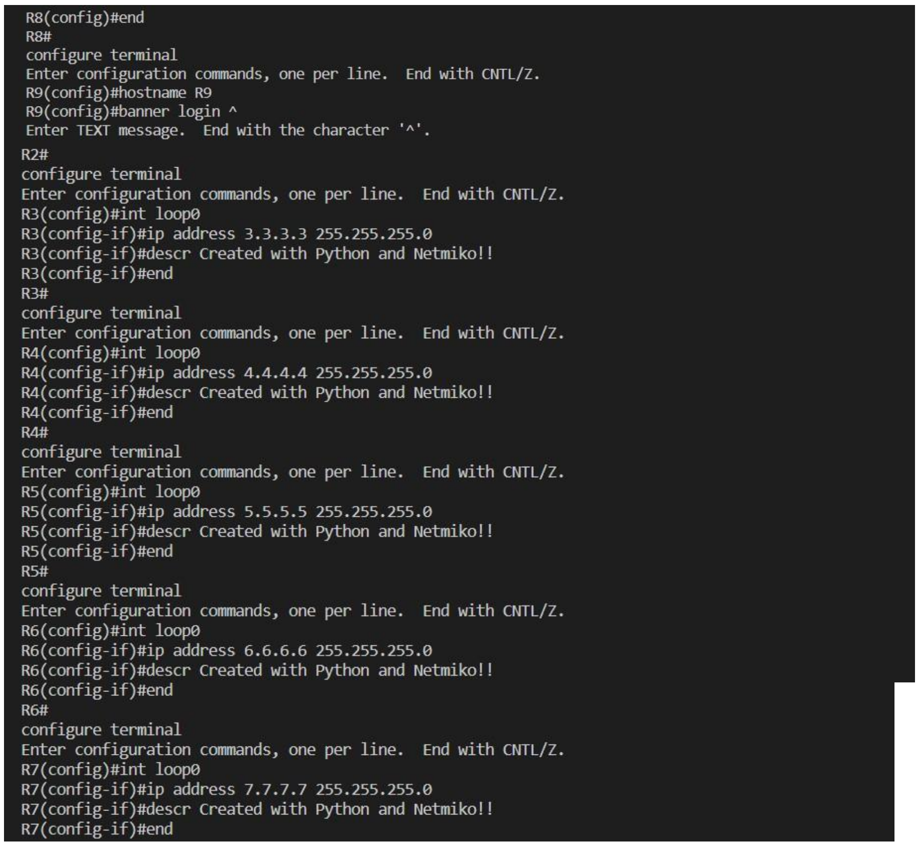


```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#hostname R1
R1(config)#banner login ^
Enter TEXT message.  End with the character '^'.
This Configuration created by Tahtihal Anher!
Happy Learning!!
^
R1(config)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#hostname R2
R2(config)#banner login ^
Enter TEXT message.  End with the character '^'.
This Configuration created by Tahtihal Anher!
Happy Learning!!
^
R2(config)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#hostname R3
R3(config)#banner login ^
Enter TEXT message.  End with the character '^'.
This Configuration created by Tahtihal Anher!
Happy Learning!!
^
R3(config)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#hostname R4
R4(config)#banner login ^
Enter TEXT message.  End with the character '^'.
```



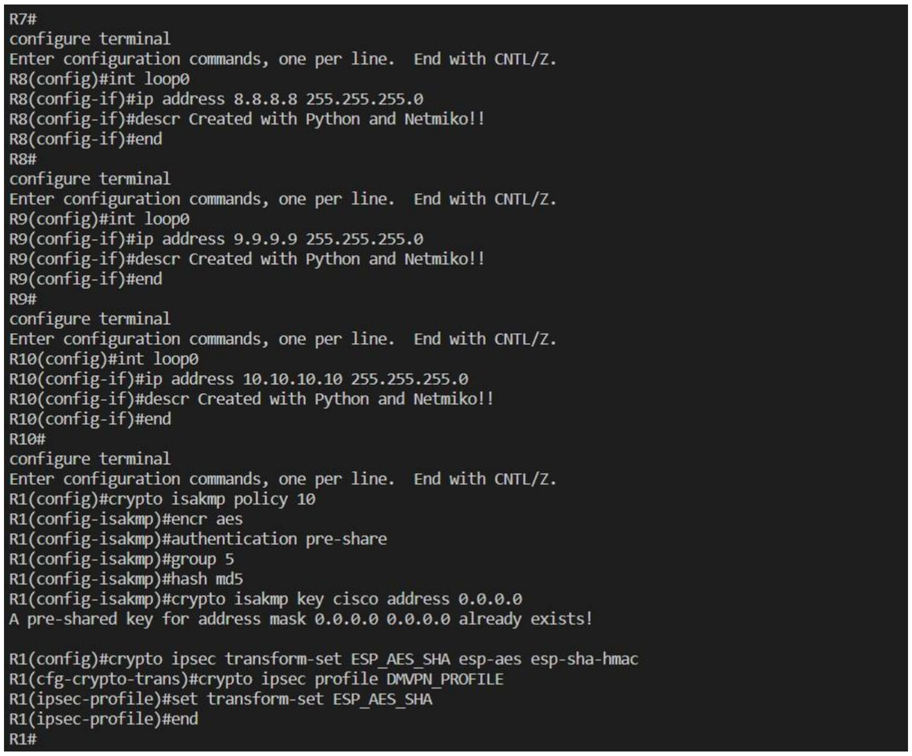
```
R4(config)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#hostname R5
R5(config)#banner login ^
Enter TEXT message.  End with the character '^'.
This Configuration created by Tahtihal Anher!
Happy Learning!!
^
R5(config)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#hostname R6
R6(config)#banner login ^
Enter TEXT message.  End with the character '^'.
This Configuration created by Tahtihal Anher!
Happy Learning!!
^
R6(config)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#hostname R7
R7(config)#banner login ^
Enter TEXT message.  End with the character '^'.
This Configuration created by Tahtihal Anher!
Happy Learning!!
^
R7(config)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#hostname R8
R8(config)#banner login ^
Enter TEXT message.  End with the character '^'.
This Configuration created by Tahtihal Anher!
Happy Learning!!
```

3.6 Setup Loopback Interfaces

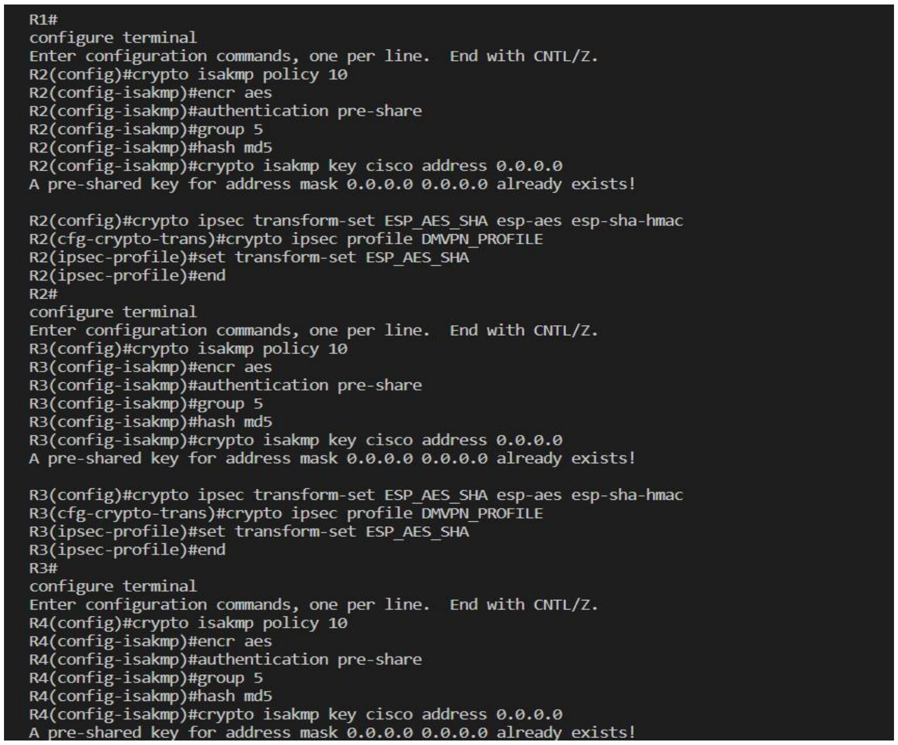


```
R8(config)#end
R8#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#hostname R9
R9(config)#banner login ^
Enter TEXT message.  End with the character '^'.
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int loop0
R3(config-if)#ip address 3.3.3.3 255.255.255.0
R3(config-if)#descr Created with Python and Netmiko!!
R3(config-if)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#int loop0
R4(config-if)#ip address 4.4.4.4 255.255.255.0
R4(config-if)#descr Created with Python and Netmiko!!
R4(config-if)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#int loop0
R5(config-if)#ip address 5.5.5.5 255.255.255.0
R5(config-if)#descr Created with Python and Netmiko!!
R5(config-if)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#int loop0
R6(config-if)#ip address 6.6.6.6 255.255.255.0
R6(config-if)#descr Created with Python and Netmiko!!
R6(config-if)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#int loop0
R7(config-if)#ip address 7.7.7.7 255.255.255.0
R7(config-if)#descr Created with Python and Netmiko!!
R7(config-if)#end
```

3.7 ISAKMP Configuration

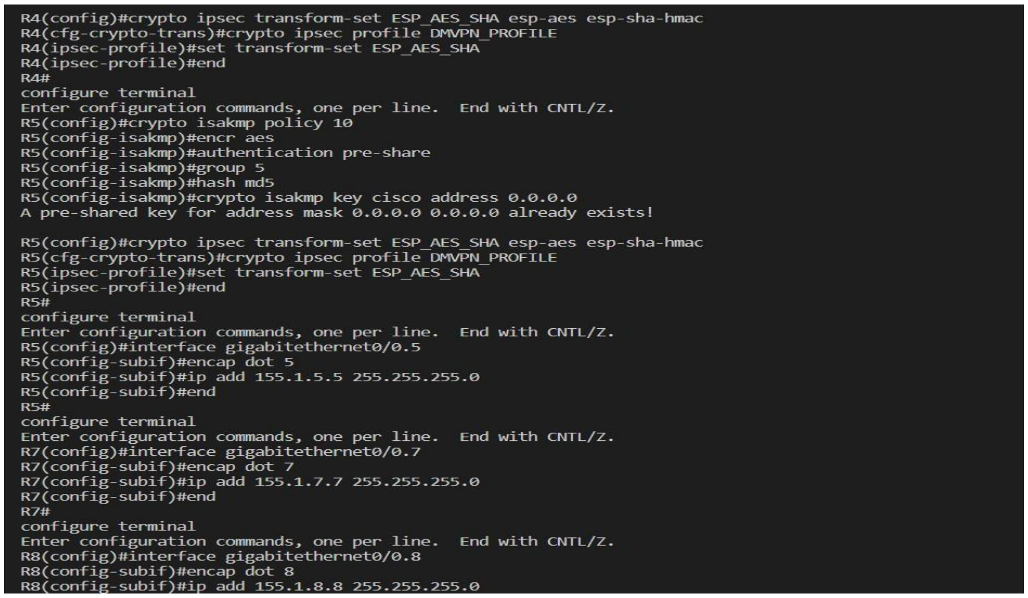


```
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#int loop0
R8(config-if)#ip address 8.8.8.8 255.255.255.0
R8(config-if)#descr Created with Python and Netmiko!!
R8(config-if)#end
R8#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#int loop0
R9(config-if)#ip address 9.9.9.9 255.255.255.0
R9(config-if)#descr Created with Python and Netmiko!!
R9(config-if)#end
R9#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R10(config)#int loop0
R10(config-if)#ip address 10.10.10.10 255.255.255.0
R10(config-if)#descr Created with Python and Netmiko!!
R10(config-if)#end
R10#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#crypto isakmp policy 10
R1(config-isakmp)#encr aes
R1(config-isakmp)#authentication pre-share
R1(config-isakmp)#group 5
R1(config-isakmp)#hash md5
R1(config-isakmp)#crypto isakmp key cisco address 0.0.0.0
A pre-shared key for address mask 0.0.0.0 0.0.0.0 already exists!

R1(config)#crypto ipsec transform-set ESP_AES_SHA esp-aes esp-sha-hmac
R1(cfg-crypto-trans)#crypto ipsec profile DMVPN_PROFILE
R1(ipsec-profile)#set transform-set ESP_AES_SHA
R1(ipsec-profile)#end
R1#
```

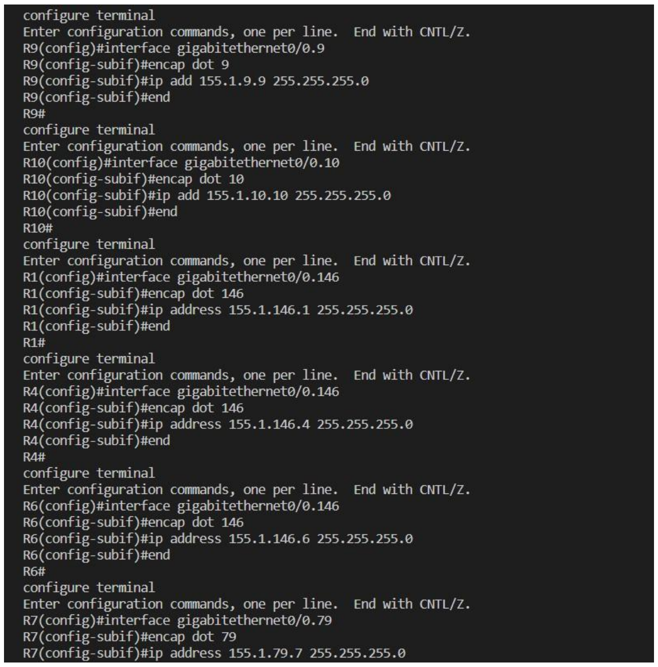


```
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#crypto isakmp policy 10
R2(config-isakmp)#encr aes
R2(config-isakmp)#authentication pre-share
R2(config-isakmp)#group 5
R2(config-isakmp)#hash md5
R2(config-isakmp)#crypto isakmp key cisco address 0.0.0.0
A pre-shared key for address mask 0.0.0.0 0.0.0.0 already exists!

R2(config)#crypto ipsec transform-set ESP_AES_SHA esp-aes esp-sha-hmac
R2(cfg-crypto-trans)#crypto ipsec profile DMVPN_PROFILE
R2(ipsec-profile)#set transform-set ESP_AES_SHA
R2(ipsec-profile)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#crypto isakmp policy 10
R3(config-isakmp)#encr aes
R3(config-isakmp)#authentication pre-share
R3(config-isakmp)#group 5
R3(config-isakmp)#hash md5
R3(config-isakmp)#crypto isakmp key cisco address 0.0.0.0
A pre-shared key for address mask 0.0.0.0 0.0.0.0 already exists!

R3(config)#crypto ipsec transform-set ESP_AES_SHA esp-aes esp-sha-hmac
R3(cfg-crypto-trans)#crypto ipsec profile DMVPN_PROFILE
R3(ipsec-profile)#set transform-set ESP_AES_SHA
R3(ipsec-profile)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#crypto isakmp policy 10
R4(config-isakmp)#encr aes
R4(config-isakmp)#authentication pre-share
R4(config-isakmp)#group 5
R4(config-isakmp)#hash md5
R4(config-isakmp)#crypto isakmp key cisco address 0.0.0.0
A pre-shared key for address mask 0.0.0.0 0.0.0.0 already exists!
```

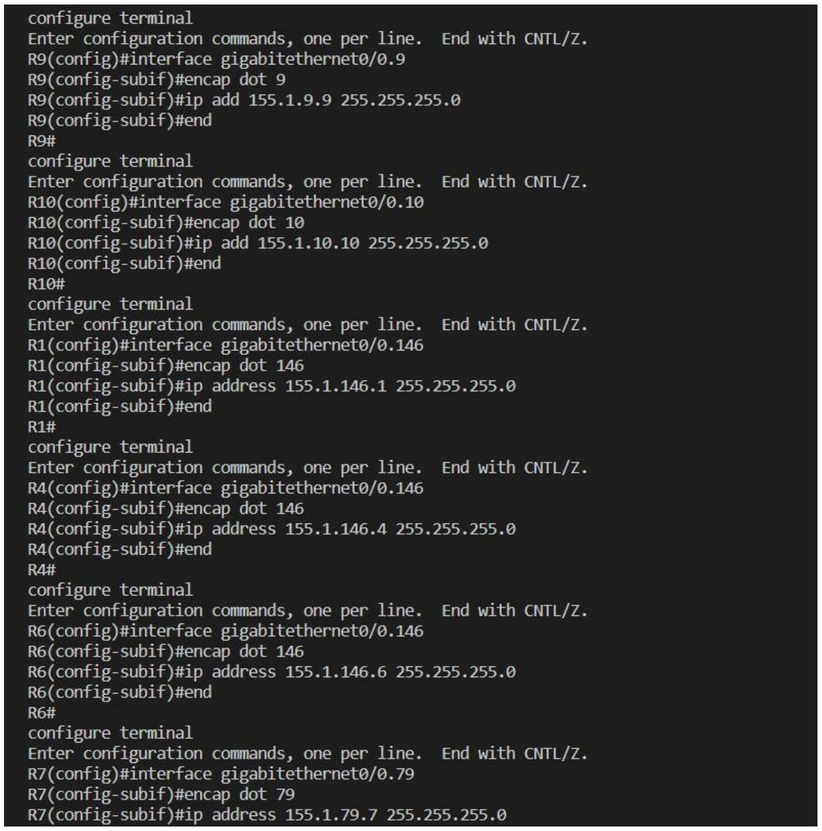
3.8 Sub interface creating for OSPF are



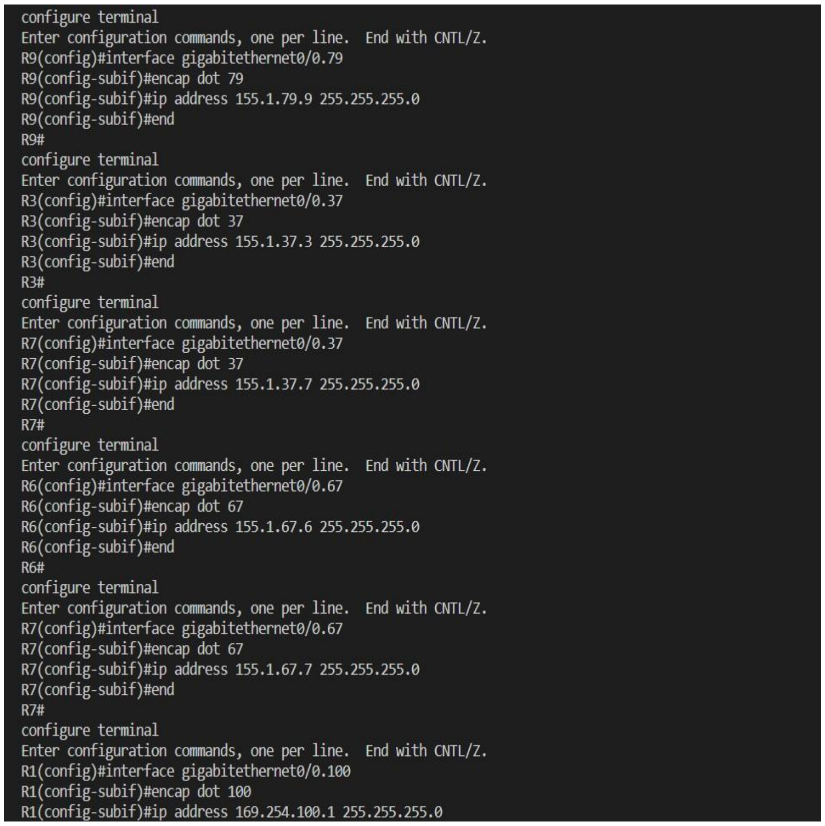
```
R4(config)#crypto ipsec transform-set ESP_AES_SHA esp-aes esp-sha-hmac
R4(cfg-crypto-trans)#crypto ipsec profile DMVPN_PROFILE
R4(ipsec-profile)#set transform-set ESP_AES_SHA
R4(ipsec-profile)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#crypto isakmp policy 10
R5(config-isakmp)#encr aes
R5(config-isakmp)#authentication pre-share
R5(config-isakmp)#group 5
R5(config-isakmp)#hash md5
R5(config-isakmp)#crypto isakmp key cisco address 0.0.0.0
A pre-shared key for address mask 0.0.0.0 0.0.0.0 already exists!

R5(config)#crypto ipsec transform-set ESP_AES_SHA esp-aes esp-sha-hmac
R5(cfg-crypto-trans)#crypto ipsec profile DMVPN_PROFILE
R5(ipsec-profile)#set transform-set ESP_AES_SHA
R5(ipsec-profile)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface gigabitethernet0/0.5
R5(config-subif)#encap dot 5
R5(config-subif)#ip add 155.1.5.5 255.255.255.0
R5(config-subif)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.7
R7(config-subif)#encap dot 7
R7(config-subif)#ip add 155.1.7.7 255.255.255.0
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#interface gigabitethernet0/0.8
R8(config-subif)#encap dot 8
R8(config-subif)#ip add 155.1.8.8 255.255.255.0
```



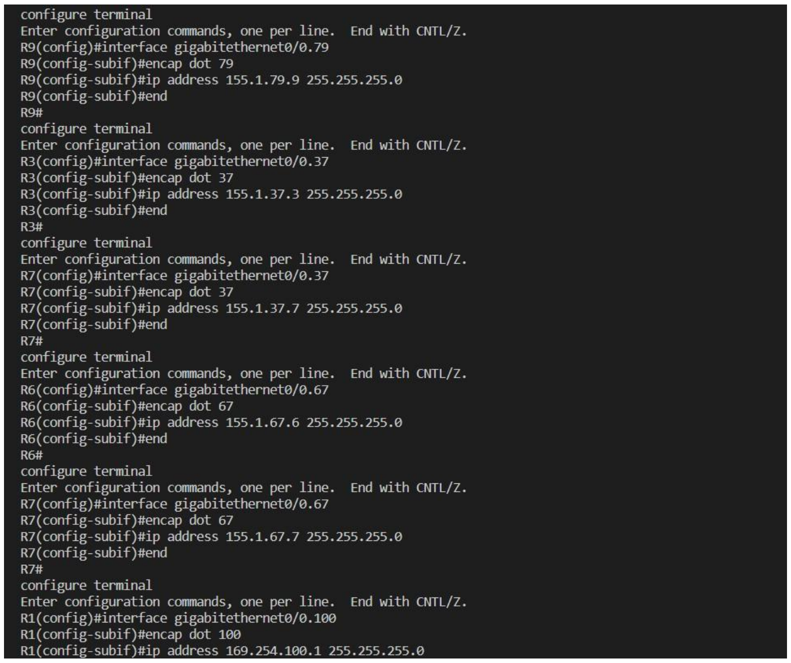
```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#interface gigabitethernet0/0.9
R9(config-subif)#encap dot 9
R9(config-subif)#ip add 155.1.9.9 255.255.255.0
R9(config-subif)#end
R9#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R10(config)#interface gigabitethernet0/0.10
R10(config-subif)#encap dot 10
R10(config-subif)#ip add 155.1.10.10 255.255.255.0
R10(config-subif)#end
R10#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface gigabitethernet0/0.146
R1(config-subif)#encap dot 146
R1(config-subif)#ip address 155.1.146.1 255.255.255.0
R1(config-subif)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface gigabitethernet0/0.146
R4(config-subif)#encap dot 146
R4(config-subif)#ip address 155.1.146.4 255.255.255.0
R4(config-subif)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#interface gigabitethernet0/0.146
R6(config-subif)#encap dot 146
R6(config-subif)#ip address 155.1.146.6 255.255.255.0
R6(config-subif)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.79
R7(config-subif)#encap dot 79
R7(config-subif)#ip address 155.1.79.7 255.255.255.0
```



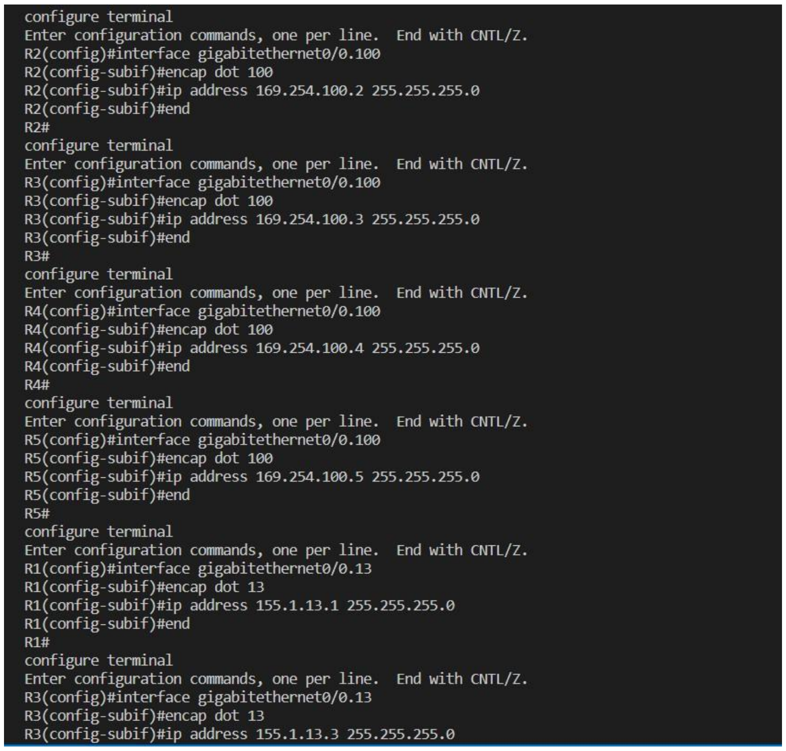
```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#interface gigabitethernet0/0.9
R9(config-subif)#encap dot 9
R9(config-subif)#ip add 155.1.9.9 255.255.255.0
R9(config-subif)#end
R9#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R10(config)#interface gigabitethernet0/0.10
R10(config-subif)#encap dot 10
R10(config-subif)#ip add 155.1.10.10 255.255.255.0
R10(config-subif)#end
R10#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface gigabitethernet0/0.146
R1(config-subif)#encap dot 146
R1(config-subif)#ip address 155.1.146.1 255.255.255.0
R1(config-subif)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface gigabitethernet0/0.146
R4(config-subif)#encap dot 146
R4(config-subif)#ip address 155.1.146.4 255.255.255.0
R4(config-subif)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#interface gigabitethernet0/0.146
R6(config-subif)#encap dot 146
R6(config-subif)#ip address 155.1.146.6 255.255.255.0
R6(config-subif)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.79
R7(config-subif)#encap dot 79
R7(config-subif)#ip address 155.1.79.7 255.255.255.0
```



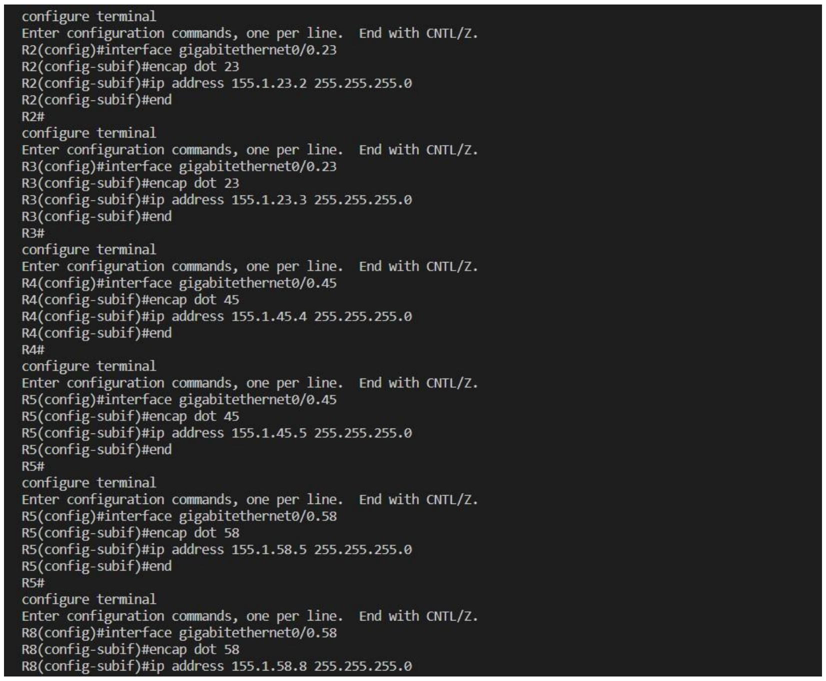
```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#interface gigabitethernet0/0.79
R9(config-subif)#encap dot 79
R9(config-subif)#ip address 155.1.79.9 255.255.255.0
R9(config-subif)#end
R9#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.37
R3(config-subif)#encap dot 37
R3(config-subif)#ip address 155.1.37.3 255.255.255.0
R3(config-subif)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.37
R7(config-subif)#encap dot 37
R7(config-subif)#ip address 155.1.37.7 255.255.255.0
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#interface gigabitethernet0/0.67
R6(config-subif)#encap dot 67
R6(config-subif)#ip address 155.1.67.6 255.255.255.0
R6(config-subif)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.67
R7(config-subif)#encap dot 67
R7(config-subif)#ip address 155.1.67.7 255.255.255.0
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface gigabitethernet0/0.100
R1(config-subif)#encap dot 100
R1(config-subif)#ip address 169.254.100.1 255.255.255.0
```



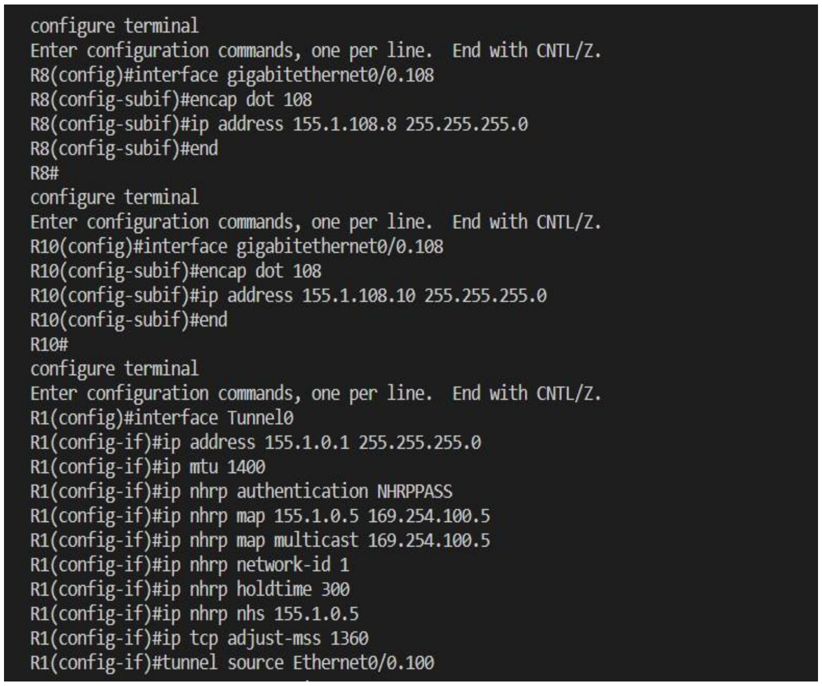
```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#interface gigabitethernet0/0.79
R9(config-subif)#encap dot 79
R9(config-subif)#ip address 155.1.79.9 255.255.255.0
R9(config-subif)#end
R9#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.37
R3(config-subif)#encap dot 37
R3(config-subif)#ip address 155.1.37.3 255.255.255.0
R3(config-subif)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.37
R7(config-subif)#encap dot 37
R7(config-subif)#ip address 155.1.37.7 255.255.255.0
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#interface gigabitethernet0/0.67
R6(config-subif)#encap dot 67
R6(config-subif)#ip address 155.1.67.6 255.255.255.0
R6(config-subif)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.67
R7(config-subif)#encap dot 67
R7(config-subif)#ip address 155.1.67.7 255.255.255.0
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface gigabitethernet0/0.100
R1(config-subif)#encap dot 100
R1(config-subif)#ip address 169.254.100.1 255.255.255.0
```

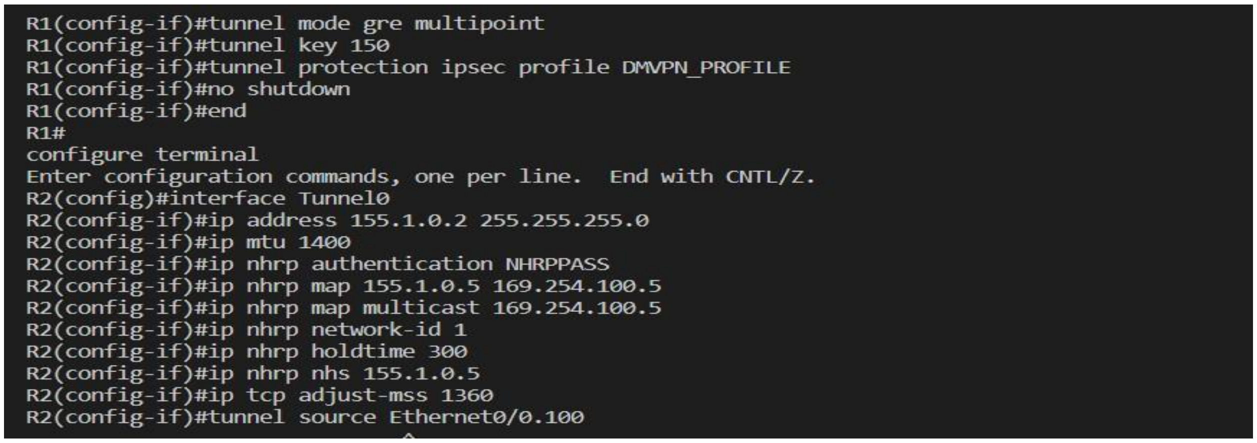


```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface gigabitethernet0/0.100
R2(config-subif)#encap dot 100
R2(config-subif)#ip address 169.254.100.2 255.255.255.0
R2(config-subif)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.100
R3(config-subif)#encap dot 100
R3(config-subif)#ip address 169.254.100.3 255.255.255.0
R3(config-subif)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface gigabitethernet0/0.100
R4(config-subif)#encap dot 100
R4(config-subif)#ip address 169.254.100.4 255.255.255.0
R4(config-subif)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface gigabitethernet0/0.100
R5(config-subif)#encap dot 100
R5(config-subif)#ip address 169.254.100.5 255.255.255.0
R5(config-subif)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface gigabitethernet0/0.13
R1(config-subif)#encap dot 13
R1(config-subif)#ip address 155.1.13.1 255.255.255.0
R1(config-subif)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.13
R3(config-subif)#encap dot 13
R3(config-subif)#ip address 155.1.13.3 255.255.255.0
```

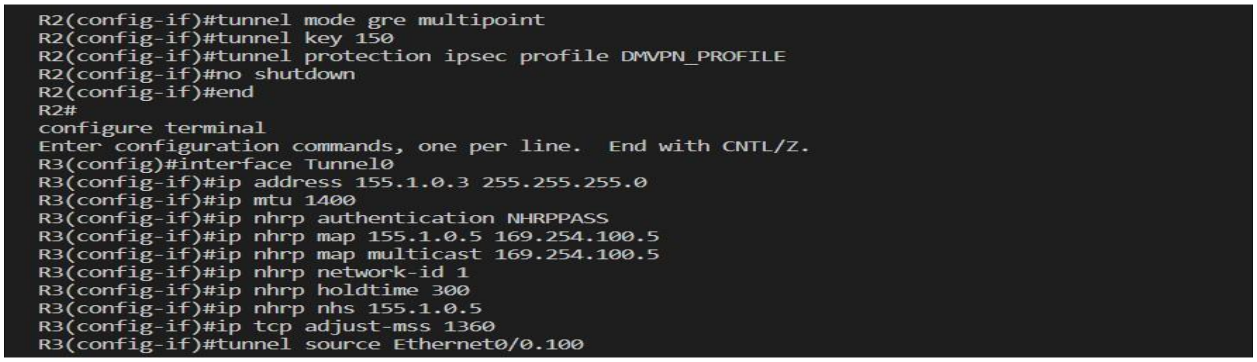

```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface gigabitethernet0/0.23
R2(config-subif)#encap dot 23
R2(config-subif)#ip address 155.1.23.2 255.255.255.0
R2(config-subif)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.23
R3(config-subif)#encap dot 23
R3(config-subif)#ip address 155.1.23.3 255.255.255.0
R3(config-subif)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface gigabitethernet0/0.45
R4(config-subif)#encap dot 45
R4(config-subif)#ip address 155.1.45.4 255.255.255.0
R4(config-subif)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface gigabitethernet0/0.45
R5(config-subif)#encap dot 45
R5(config-subif)#ip address 155.1.45.5 255.255.255.0
R5(config-subif)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface gigabitethernet0/0.58
R5(config-subif)#encap dot 58
R5(config-subif)#ip address 155.1.58.5 255.255.255.0
R5(config-subif)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#interface gigabitethernet0/0.58
R8(config-subif)#encap dot 58
R8(config-subif)#ip address 155.1.58.8 255.255.255.0
```

3.9 Tunnel Router Configuration with its interfaces

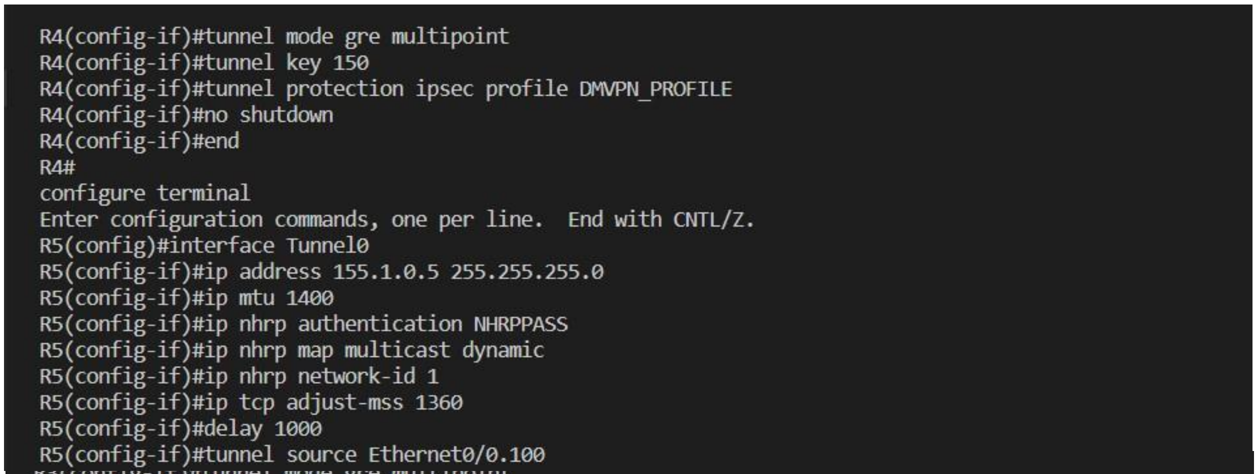

```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#interface gigabitethernet0/0.108
R8(config-subif)#encap dot 108
R8(config-subif)#ip address 155.1.108.8 255.255.255.0
R8(config-subif)#end
R8#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R10(config)#interface gigabitethernet0/0.108
R10(config-subif)#encap dot 108
R10(config-subif)#ip address 155.1.108.10 255.255.255.0
R10(config-subif)#end
R10#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface Tunnel0
R1(config-if)#ip address 155.1.0.1 255.255.255.0
R1(config-if)#ip mtu 1400
R1(config-if)#ip nhrp authentication NHRPPASS
R1(config-if)#ip nhrp map 155.1.0.5 169.254.100.5
R1(config-if)#ip nhrp map multicast 169.254.100.5
R1(config-if)#ip nhrp network-id 1
R1(config-if)#ip nhrp holdtime 300
R1(config-if)#ip nhrp nhs 155.1.0.5
R1(config-if)#ip tcp adjust-mss 1360
R1(config-if)#tunnel source Ethernet0/0.100
```



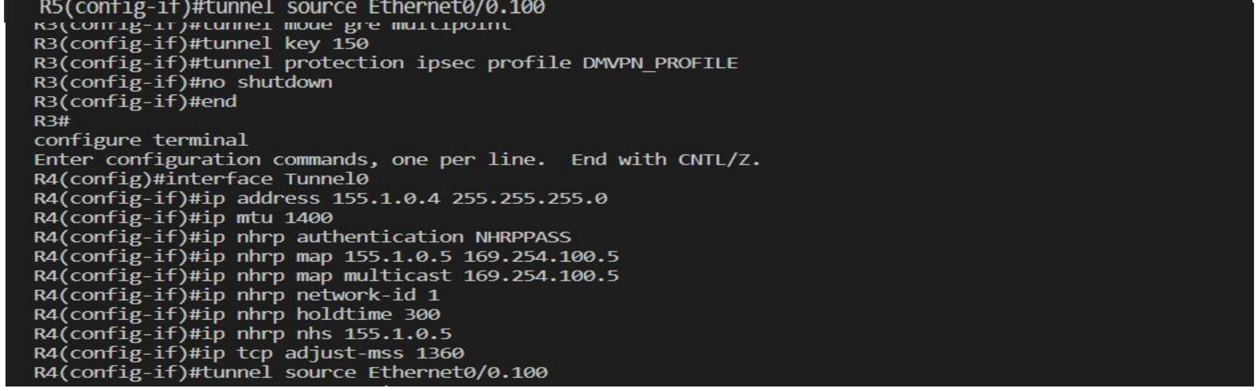
```
R1(config-if)#tunnel mode gre multipoint
R1(config-if)#tunnel key 150
R1(config-if)#tunnel protection ipsec profile DMVPN_PROFILE
R1(config-if)#no shutdown
R1(config-if)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface Tunnel0
R2(config-if)#ip address 155.1.0.2 255.255.255.0
R2(config-if)#ip mtu 1400
R2(config-if)#ip nhrp authentication NHRPPASS
R2(config-if)#ip nhrp map 155.1.0.5 169.254.100.5
R2(config-if)#ip nhrp map multicast 169.254.100.5
R2(config-if)#ip nhrp network-id 1
R2(config-if)#ip nhrp holdtime 300
R2(config-if)#ip nhrp nhs 155.1.0.5
R2(config-if)#ip tcp adjust-mss 1360
R2(config-if)#tunnel source Ethernet0/0.100
```



```
R2(config-if)#tunnel mode gre multipoint
R2(config-if)#tunnel key 150
R2(config-if)#tunnel protection ipsec profile DMVPN_PROFILE
R2(config-if)#no shutdown
R2(config-if)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface Tunnel0
R3(config-if)#ip address 155.1.0.3 255.255.255.0
R3(config-if)#ip mtu 1400
R3(config-if)#ip nhrp authentication NHRPPASS
R3(config-if)#ip nhrp map 155.1.0.5 169.254.100.5
R3(config-if)#ip nhrp map multicast 169.254.100.5
R3(config-if)#ip nhrp network-id 1
R3(config-if)#ip nhrp holdtime 300
R3(config-if)#ip nhrp nhs 155.1.0.5
R3(config-if)#ip tcp adjust-mss 1360
R3(config-if)#tunnel source Ethernet0/0.100
```

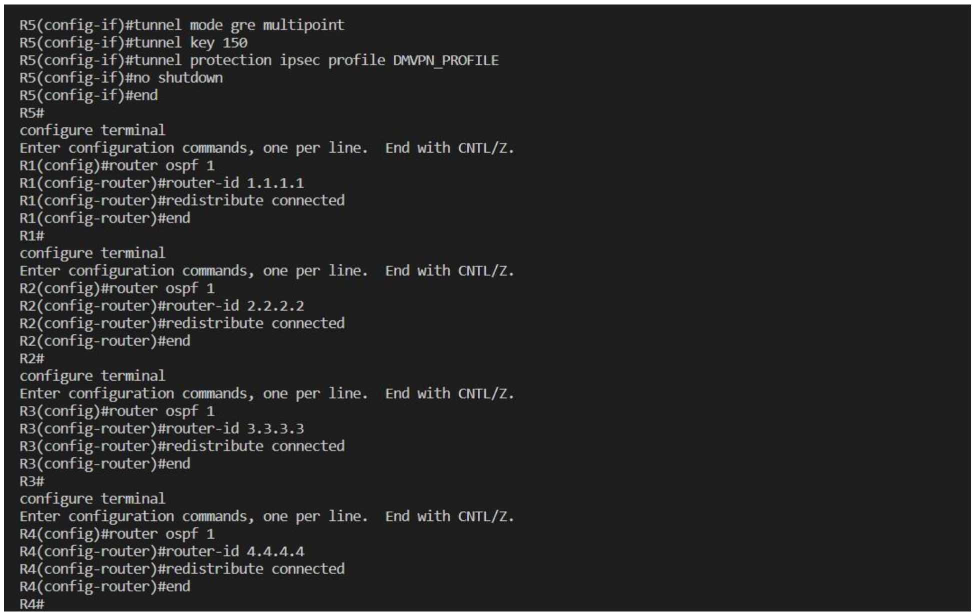


```
R4(config-if)#tunnel mode gre multipoint
R4(config-if)#tunnel key 150
R4(config-if)#tunnel protection ipsec profile DMVPN_PROFILE
R4(config-if)#no shutdown
R4(config-if)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface Tunnel0
R5(config-if)#ip address 155.1.0.5 255.255.255.0
R5(config-if)#ip mtu 1400
R5(config-if)#ip nhrp authentication NHRPPASS
R5(config-if)#ip nhrp map multicast dynamic
R5(config-if)#ip nhrp network-id 1
R5(config-if)#ip tcp adjust-mss 1360
R5(config-if)#delay 1000
R5(config-if)#tunnel source Ethernet0/0.100
```

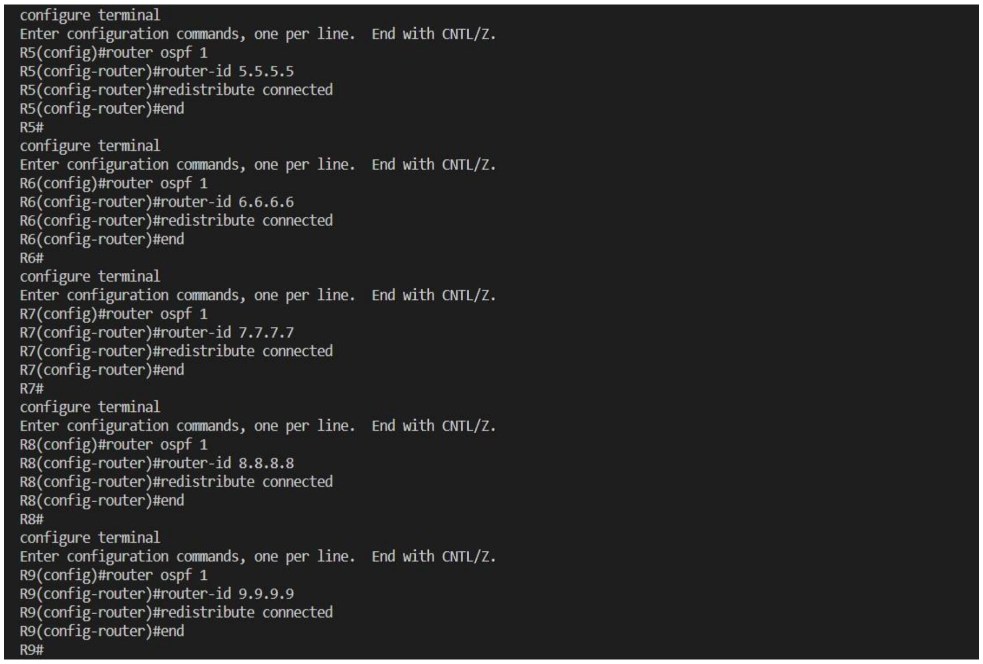


```
R3(config-if)#tunnel mode gre multipoint
R3(config-if)#tunnel key 150
R3(config-if)#tunnel protection ipsec profile DMVPN_PROFILE
R3(config-if)#no shutdown
R3(config-if)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface Tunnel0
R4(config-if)#ip address 155.1.0.4 255.255.255.0
R4(config-if)#ip mtu 1400
R4(config-if)#ip nhrp authentication NHRPPASS
R4(config-if)#ip nhrp map 155.1.0.5 169.254.100.5
R4(config-if)#ip nhrp map multicast 169.254.100.5
R4(config-if)#ip nhrp network-id 1
R4(config-if)#ip nhrp holdtime 300
R4(config-if)#ip nhrp nhs 155.1.0.5
R4(config-if)#ip tcp adjust-mss 1360
R4(config-if)#tunnel source Ethernet0/0.100
```

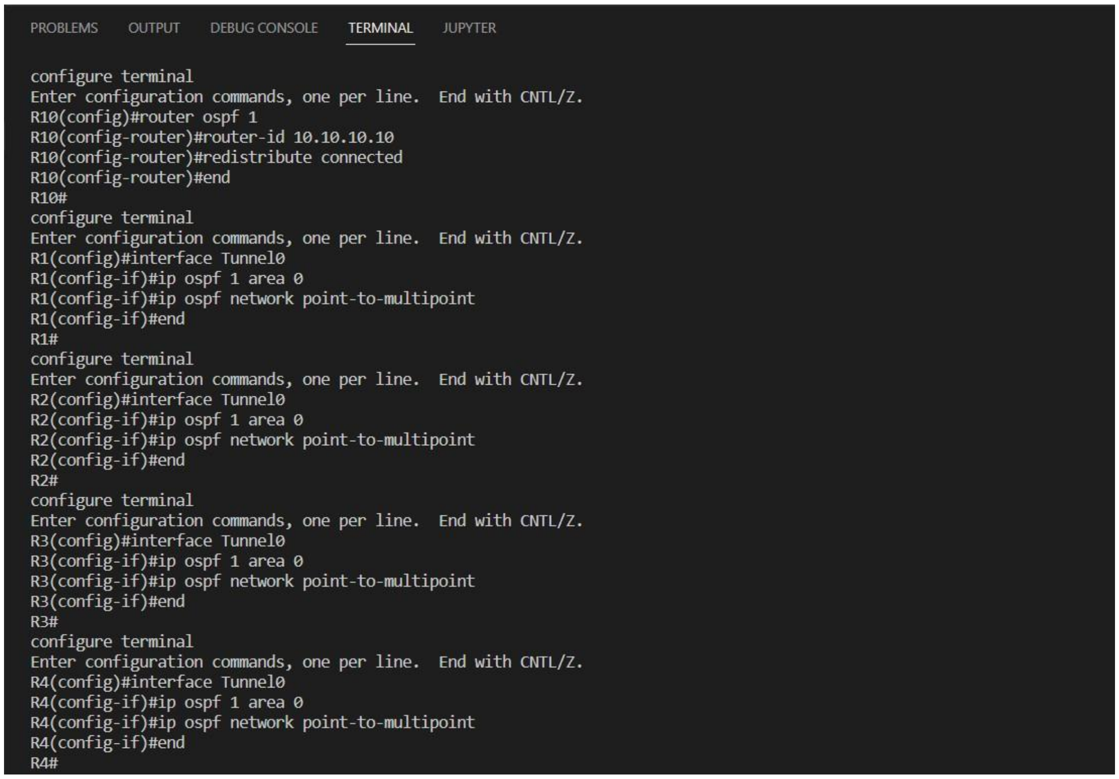
3.10 Routing Protocol OSPF Configuration



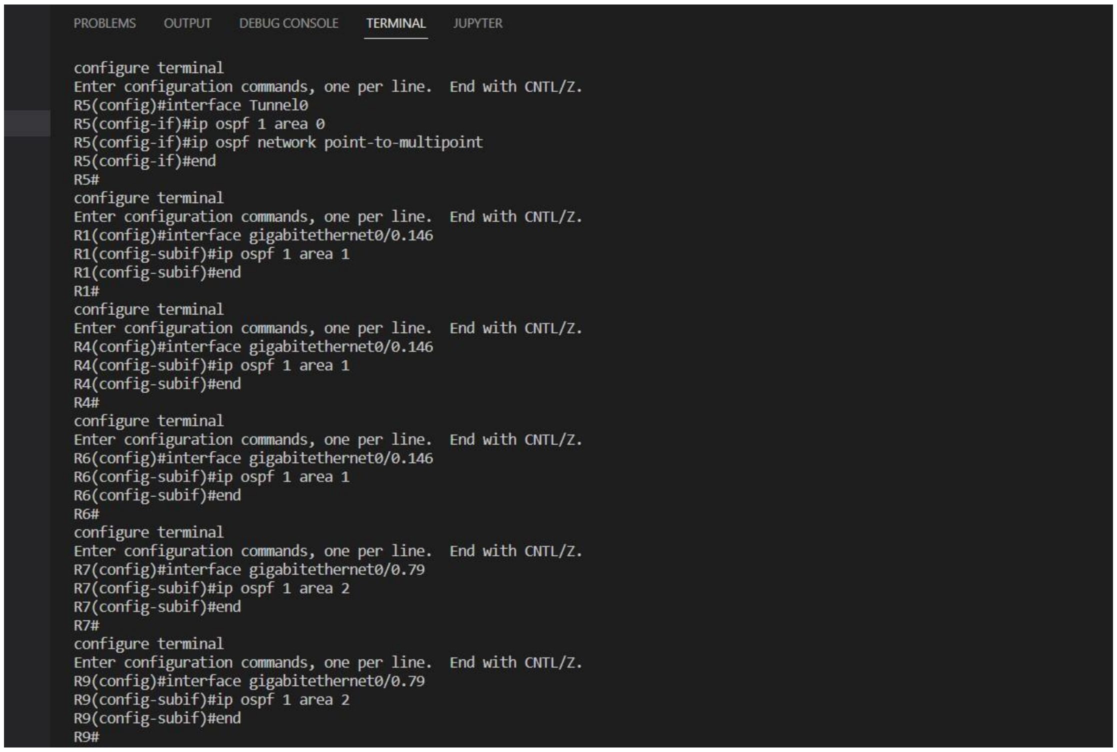
```
R5(config-if)#tunnel mode gre multipoint
R5(config-if)#tunnel key 150
R5(config-if)#tunnel protection ipsec profile DMVPN_PROFILE
R5(config-if)#no shutdown
R5(config-if)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#router ospf 1
R1(config-router)#router-id 1.1.1.1
R1(config-router)#redistribute connected
R1(config-router)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#router ospf 1
R2(config-router)#router-id 2.2.2.2
R2(config-router)#redistribute connected
R2(config-router)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#router ospf 1
R3(config-router)#router-id 3.3.3.3
R3(config-router)#redistribute connected
R3(config-router)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#router ospf 1
R4(config-router)#router-id 4.4.4.4
R4(config-router)#redistribute connected
R4(config-router)#end
R4#
```



```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#router ospf 1
R5(config-router)#router-id 5.5.5.5
R5(config-router)#redistribute connected
R5(config-router)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#router ospf 1
R6(config-router)#router-id 6.6.6.6
R6(config-router)#redistribute connected
R6(config-router)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#router ospf 1
R7(config-router)#router-id 7.7.7.7
R7(config-router)#redistribute connected
R7(config-router)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#router ospf 1
R8(config-router)#router-id 8.8.8.8
R8(config-router)#redistribute connected
R8(config-router)#end
R8#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#router ospf 1
R9(config-router)#router-id 9.9.9.9
R9(config-router)#redistribute connected
R9(config-router)#end
R9#
```

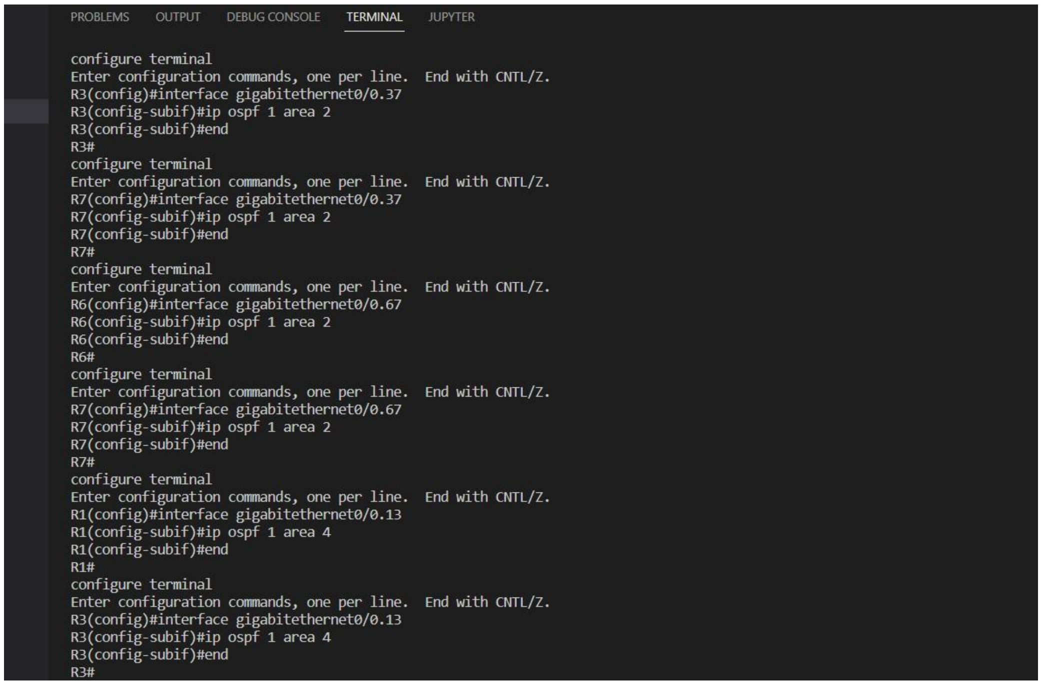


```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R10(config)#router ospf 1
R10(config-router)#router-id 10.10.10.10
R10(config-router)#redistribute connected
R10(config-router)#end
R10#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface Tunnel0
R1(config-if)#ip ospf 1 area 0
R1(config-if)#ip ospf network point-to-multipoint
R1(config-if)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface Tunnel0
R2(config-if)#ip ospf 1 area 0
R2(config-if)#ip ospf network point-to-multipoint
R2(config-if)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface Tunnel0
R3(config-if)#ip ospf 1 area 0
R3(config-if)#ip ospf network point-to-multipoint
R3(config-if)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface Tunnel0
R4(config-if)#ip ospf 1 area 0
R4(config-if)#ip ospf network point-to-multipoint
R4(config-if)#end
R4#
```

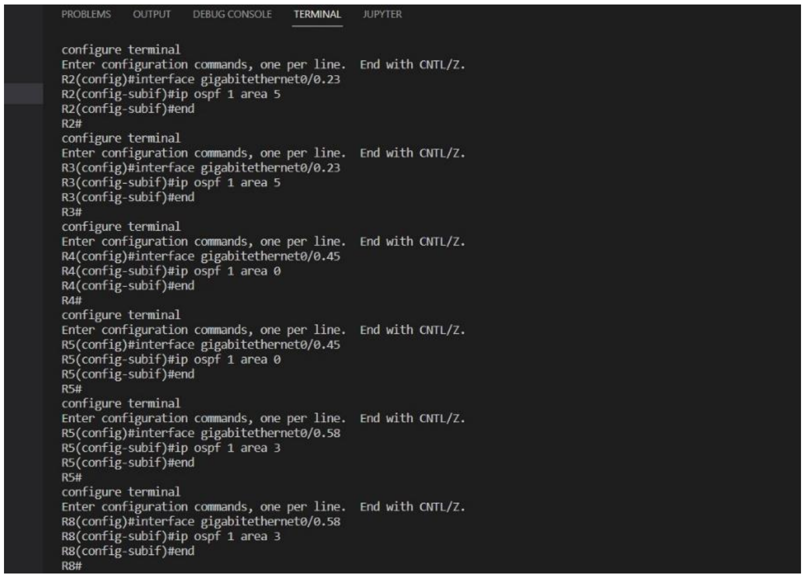


```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface Tunnel0
R5(config-if)#ip ospf 1 area 0
R5(config-if)#ip ospf network point-to-multipoint
R5(config-if)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface gigabitethernet0/0.146
R1(config-subif)#ip ospf 1 area 1
R1(config-subif)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface gigabitethernet0/0.146
R4(config-subif)#ip ospf 1 area 1
R4(config-subif)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#interface gigabitethernet0/0.146
R6(config-subif)#ip ospf 1 area 1
R6(config-subif)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.79
R7(config-subif)#ip ospf 1 area 2
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R9(config)#interface gigabitethernet0/0.79
R9(config-subif)#ip ospf 1 area 2
R9(config-subif)#end
R9#
```

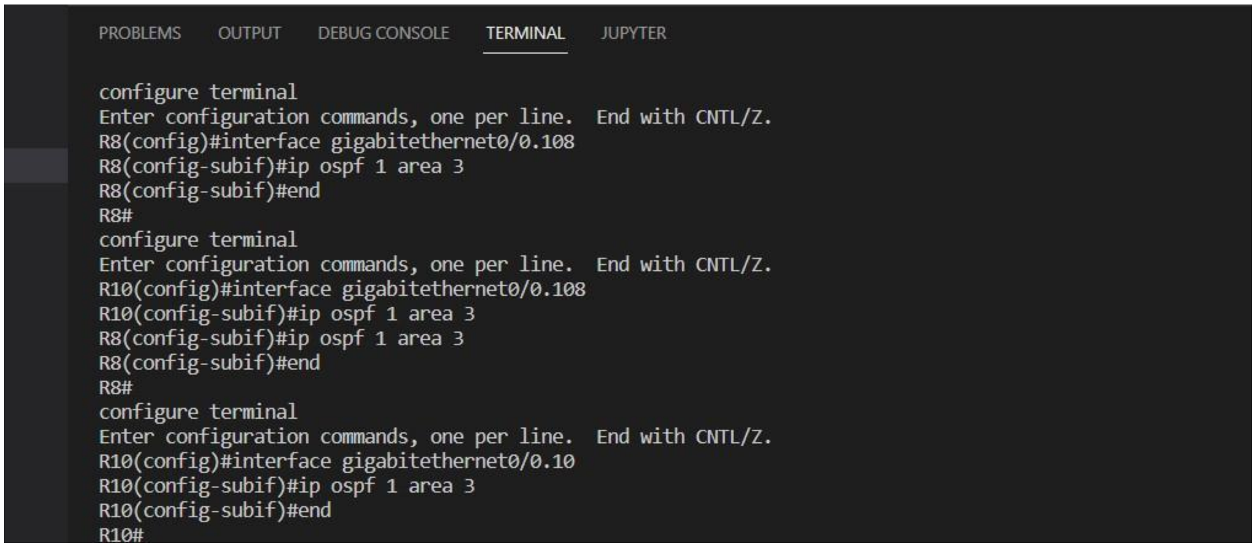
3.11OSPF area assigned by sub interface



```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.37
R3(config-subif)#ip ospf 1 area 2
R3(config-subif)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.37
R7(config-subif)#ip ospf 1 area 2
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#interface gigabitethernet0/0.67
R6(config-subif)#ip ospf 1 area 2
R6(config-subif)#end
R6#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#interface gigabitethernet0/0.67
R7(config-subif)#ip ospf 1 area 2
R7(config-subif)#end
R7#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface gigabitethernet0/0.13
R1(config-subif)#ip ospf 1 area 4
R1(config-subif)#end
R1#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.13
R3(config-subif)#ip ospf 1 area 4
R3(config-subif)#end
R3#
```

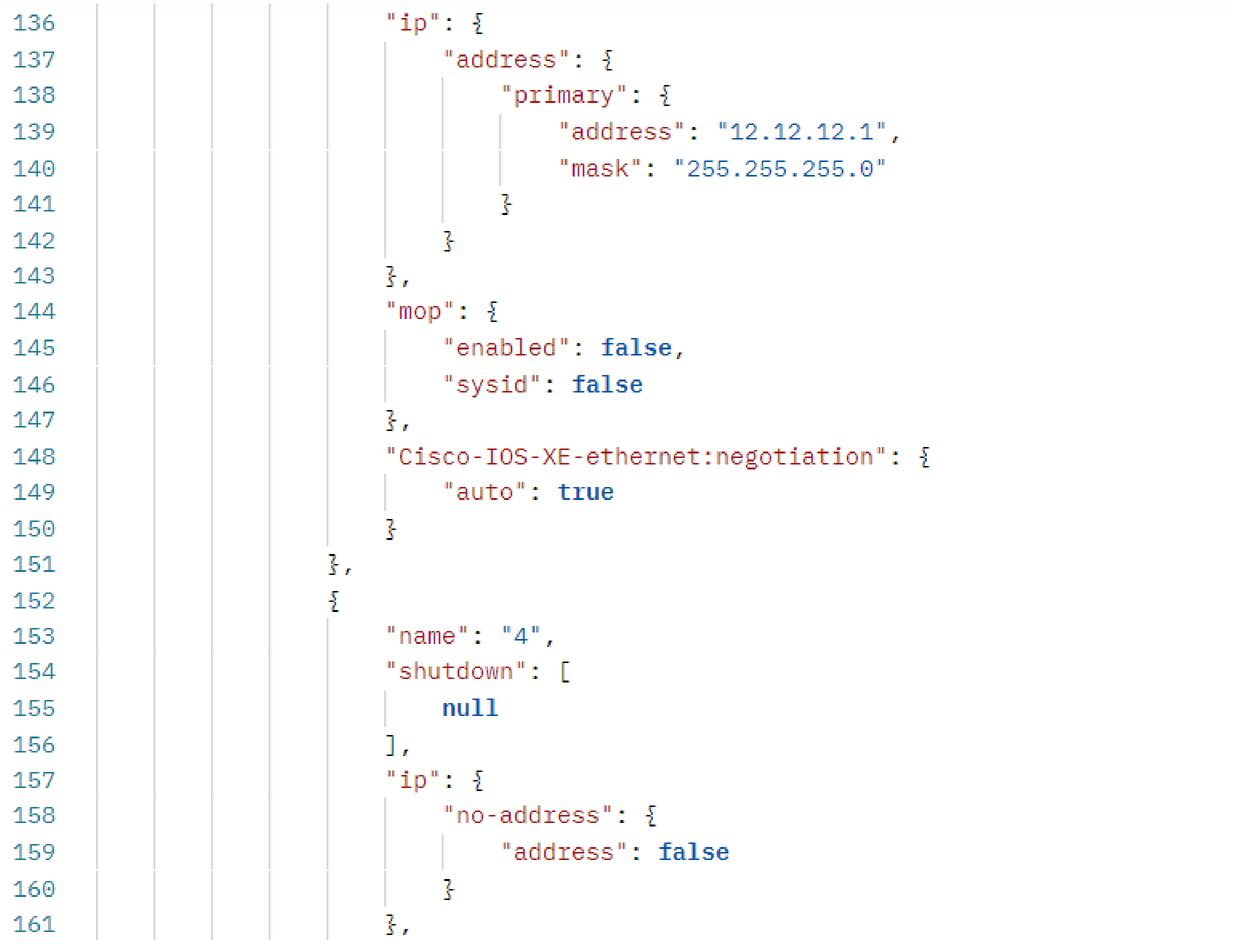


```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface gigabitethernet0/0.23
R2(config-subif)#ip ospf 1 area 5
R2(config-subif)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface gigabitethernet0/0.23
R3(config-subif)#ip ospf 1 area 5
R3(config-subif)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#interface gigabitethernet0/0.45
R4(config-subif)#ip ospf 1 area 0
R4(config-subif)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface gigabitethernet0/0.45
R5(config-subif)#ip ospf 1 area 0
R5(config-subif)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R5(config)#interface gigabitethernet0/0.58
R5(config-subif)#ip ospf 1 area 3
R5(config-subif)#end
R5#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#interface gigabitethernet0/0.58
R8(config-subif)#ip ospf 1 area 3
R8(config-subif)#end
R8#
```

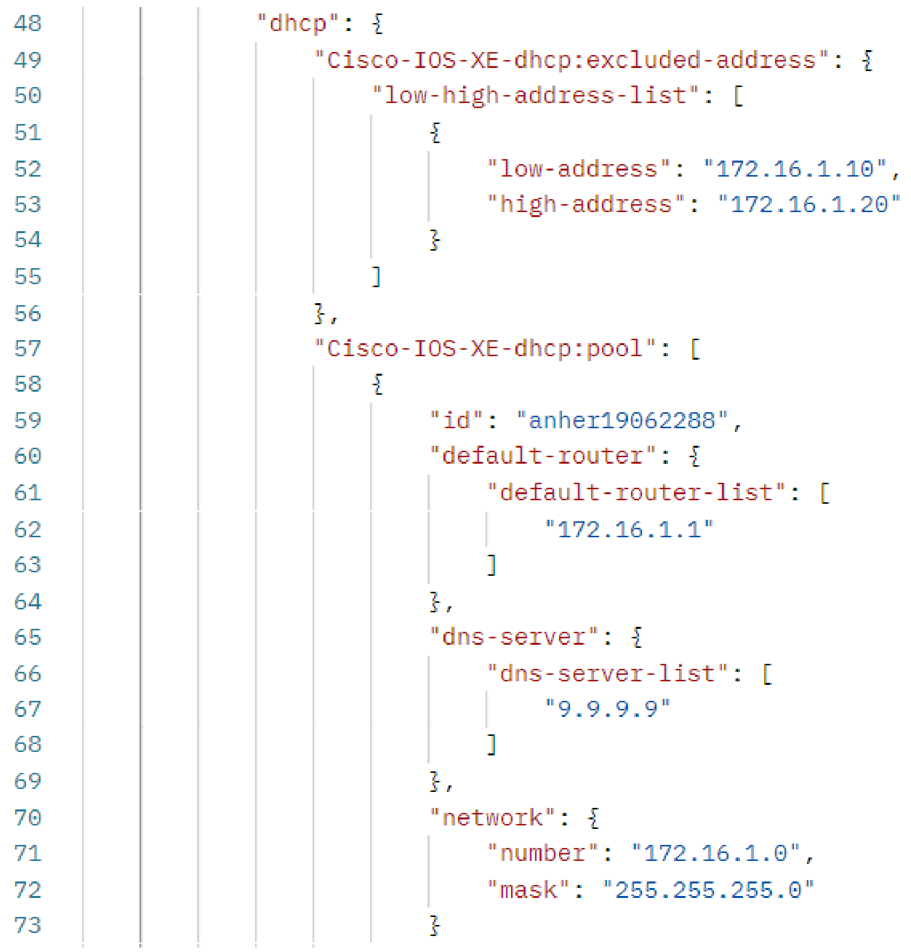


```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R8(config)#interface gigabitethernet0/0.108
R8(config-subif)#ip ospf 1 area 3
R8(config-subif)#end
R8#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R10(config)#interface gigabitethernet0/0.108
R10(config-subif)#ip ospf 1 area 3
R8(config-subif)#ip ospf 1 area 3
R8(config-subif)#end
R8#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R10(config)#interface gigabitethernet0/0.10
R10(config-subif)#ip ospf 1 area 3
R10(config-subif)#end
R10#
```

4.0 RESTCONF OUTPUT as JSON FORMAT



```
                    "ip": {
                        "address": {
                            "primary": {
                                "address": "12.12.12.1",
                                "mask": "255.255.255.0"
                            }
                        }
                    },
                    "mop": {
                        "enabled": false,
                        "sysid": false
                    },
                    "Cisco-IOS-XE-ethernet:negotiation": {
                        "auto": true
                    }
                },
                {
                    "name": "4",
                    "shutdown": [
                        null
                    ],
                    "ip": {
                        "no-address": {
                            "address": false
                        }
                    },
```



```
            "dhcp": {
                "Cisco-IOS-XE-dhcp:excluded-address": {
                    "low-high-address-list": [
                        {
                            "low-address": "172.16.1.10",
                            "high-address": "172.16.1.20"
                        }
                    ]
                },
                "Cisco-IOS-XE-dhcp:pool": [
                    {
                        "id": "anher19062288",
                        "default-router": {
                            "default-router-list": [
                                "172.16.1.1"
                            ]
                        },
                        "dns-server": {
                            "dns-server-list": [
                                "9.9.9.9"
                            ]
                        },
                        "network": {
                            "number": "172.16.1.0",
                            "mask": "255.255.255.0"
                        }
```

Thank You Sir For Your Time and Consideration.