Journal of Computing Research and Innovation (JCRINN) Vol. 6 No. 4 (2021)
https://jcrinn.com : eISSN: 2600-8793
https://doi.org/10.24191/jcrinn.v6i4.237

Network Automation using Ansible for EIGRP Network

Mohd Faris Mohd Fuzi[1]*, Khairunnisa Abdullah[2], Iman Hazwam Abd Halim[3], Rafiza Ruslan[4]
[1,2,3,4] Faculty of Computer and Mathematical Sciences,
Universiti Teknologi MARA Perlis Branch, Arau Campus, 02600 Arau, Perlis, Malaysia

*Corresponding author: * farisfuzi@uitm.edu.my*
Received Date: 15 August 2021
Accepted Date: 4 September 2021
Revised date: 28 September 2021
Published Date: 1 October 2021

HIGHLIGHTS

- All the scripts are configured on the Network Automation using Ansible Playbook.
- YAML file has some rules that need to be followed to make the script run.
- Develop scripting to push the EIGRP routing protocol and EIGRP advanced configuration from the Network Automation Docker to the three routers in the GNS3 environment.
- The scripting has successfully automated the configuration and verified it in the testing phase.

ABSTRACT

Network automation has evolved into a solution that emphasizes efficiency in all areas. Furthermore, communication and computer networks rely on a platform that provides the necessary technological infrastructure for packet transfer through the Internet using routing protocols. The Enhanced Interior Gateway Routing Protocol (EIGRP) is a hybrid routing protocol that combines the properties of both distance-vector and link-state routing methods. The traditional technique to configure EIGRP is inefficient and requires repeated processes compared to the network automation concept. Network automation helps to assist network administrators in automating and verifying the EIGRP configuration using scripting. This paper implemented network automation using Ansible to configure EIGRP routing and advanced configuration in the GNS3 environment. This study is focused on automated scripting to configure IP Addresses to the interfaces, EIGRP routing protocol, a default static route and advanced EIGRP configurations. Ansible ran the scripting on Network Automation Docker and pushed the configurations to the routers. The network automation docker communicated with other routers via SSH. In the testing phase, the running configuration between the traditional approach and automation scripting in the Ansible playbook was compared to verify EIGRP configurations' accuracy. The findings show that Ansible has successfully deployed the configuration to the routers with no errors. Ansible can help network administrators minimized human mistakes, reduce time-consuming and enable device visibility across the network environment. Implementing EIGRP authentication and hardening process can enhance the network security level for future study.

Keywords: *Network Automation, Ansible, GNS3, EIGRP*

INTRODUCTION

Network automation has become a solution provided today that prioritizes effectiveness in every area. The concept of network automation focuses on automating the management, deployment, testing, and activity of physical devices or virtual networks in a control node with a running program. It assists in performing several tasks, reducing time consumption, and eliminating potential mistakes since all configurations are written and served in a script (Islami, Musa & Lamsani, 2020). Besides that, network automation is faster and efficient than traditional operations (Mazin, Rahman, Kassim & Mahmud, 2020). All the tasks and processes can be scripted automatically. Network configurations have been simplified into a file. The file used executes repetitive tasks and arranges the operation sequentially (Islami et al., 2020). The network automation is implemented using a tool called Ansible. Ansible is agent-less which does not require additional software installed and communication via SSH (Shah, Dubaria, & Widhalm, 2018). The Internet has revolutionized the way in communication networks. Communication and computer networks rely on a platform that enables the technological infrastructure to transmit packets over the Internet using routing protocols. Enhanced Interior Gateway Routing Protocol (EIGRP) is based on an advanced distance vector (Biradar, 2020). EIGRP disseminates network topology to the adjacent routers and determines the best path using the distance vector and link-state algorithm. The metrics used in the EIGRP routing protocol are delay and bandwidth (Masruroh, Robby, & Hakiem, 2016). EIGRP has to establish a neighbor relationship before the routing updates are sent and support Variable-Length Subnet Masking (VLSM) (Goyal, 2018).

RELATED WORKS

Network Automation

Network Automation improves the efficiency of configuration network devices using automated scripting rather than the traditional method. Mazin et al. (2020) studied the performance differences in time between manual configuration and automation using scripting. There were 36 Cisco network devices used in the research with various types of IOS image versions. The study found that implementing automation to configure the devices improved efficiency and reduced the time needed. The results show that the time required is faster and efficient when using automation (120 seconds) than the manual configuration (5797 seconds).

Islami et al. (2020) studied the implementation using network automation in Raspberry Pi to configure network devices. The study used the Ansible tool to complete the tasks. The result concluded that network automation could reduce equipment configuration and maintenance time and reduce human error in configuration syntaxes. Their study indicated that Raspberry Pi has a restriction with the use of Ansible version.

Based on the research conducted by Wijaya, J. (2018), the study showed scripting effectiveness in implementing network devices. The method used in this study is by using Ansible as an automation tool to configure the network device, the Ubuntu environment and the CISCO IOS image. However, Ansible supports only Linux and not Windows environments. The study concluded that the network administrator must create a proper infrastructure and implement automation scripting using Ansible without configuring each device.

Ortiz-Garces, Echeverria, and Andrade (2021) proposed a network automation model to harden the campus network. The model consists of three phases focused on the communication protocols, hardening configurations and playbook deployment. The researchers implemented a network automation model using Ansible and Open Shortest Path First (OPSF) for the routing protocol. The hardening process has two levels covered on the rules: network AAA rules, access rules, routing rules and OSPF authentication. Their study found an increment in the percentage of hardening of the campus network.

Routing Protocol

Okonkwo and Emmanuel (2020) determined a comparative study of RIP, OSPF, and EIGRP using ring topologies on the GNS3 network emulator. Their study involved designing four, six, eight, and twenty routers connected using star and mesh topologies. The researchers configured EIGRP and OSPF routing protocol using a network simulator and Cisco hardware equipment. The limitation found was that it had been restricted to several network routers. Then, in convergence length, EIGRP had higher efficiency, the period when a connection fails and added new links to the network rather than the OSPF protocol for routing.

Manzoor, Hussain, and Mehrban (2020) studied the best path to each network connection. The analysis made with routing protocols EIGRP, OSPF, and BGP are used in this topology and configured route distribution on these routers. Different types of data traffic are generated for network convergence, throughput, and packet delay. However, the limitation is that EIGRP has been used in a small environment. From the study, EIGRP is better in convergence while OSPF is better in packet delay. There are some similarities between their project and this project. This project focuses on the configuration of EIGRP using Network Automation.

Graphical Network Simulator-3 (GNS3)

Mihaila, Balan, Curpen, and Sandu (2017) discussed how to show scripting effectiveness in implementing network devices. It emulated the network topology using GNS3, Ubuntu Docker Container as a central feature and controlled the network devices using Paramiko and Netmiko. This study stated that the new programmatic method is supported only by more recent devices. The finding from this study shows that the controllability of the network is extremely easy, and modification could be implemented faster. The project is quite similar, which uses the GNS3 emulator platform but different network automation tools.

METHODOLOGY

Design and Development

The design and development phase begins with all the required software installed to build the network topology in the GNS3 interface. The implementation starts with an IOS image loaded into the GNS3. Then, the topology is created and evaluated. If the topology is appropriate, the automation script is configured in Ansible. If not, the topology has to be modified. Next, the scripting was loaded into the devices and tested to check if the configurations were functioning. It also has to be changed if the network is not successfully operating. Figure 1 illustrates the network topology of this project.

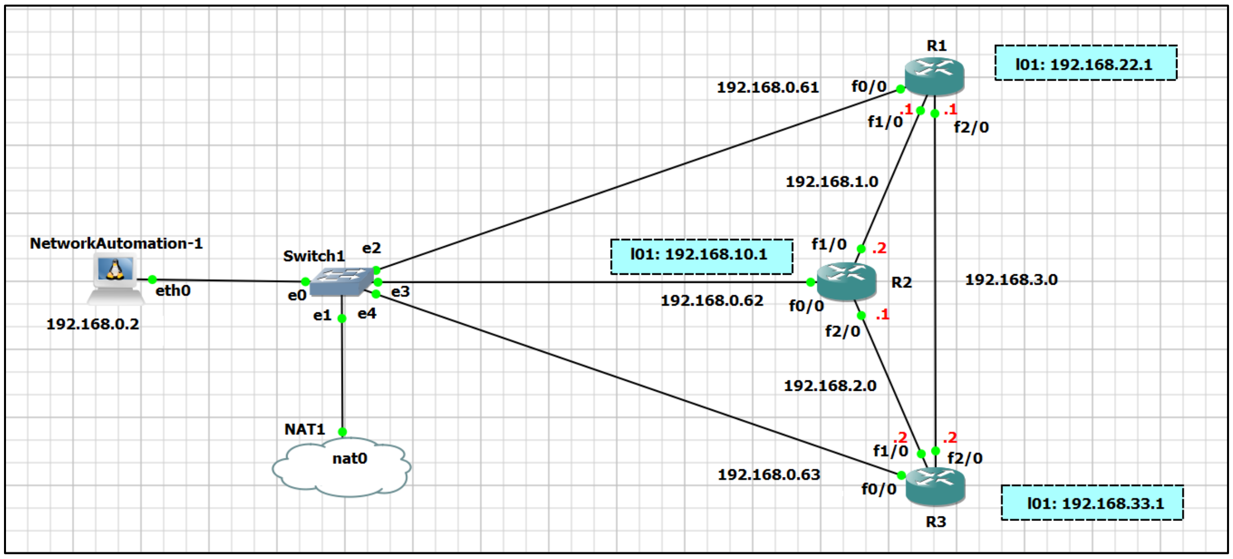
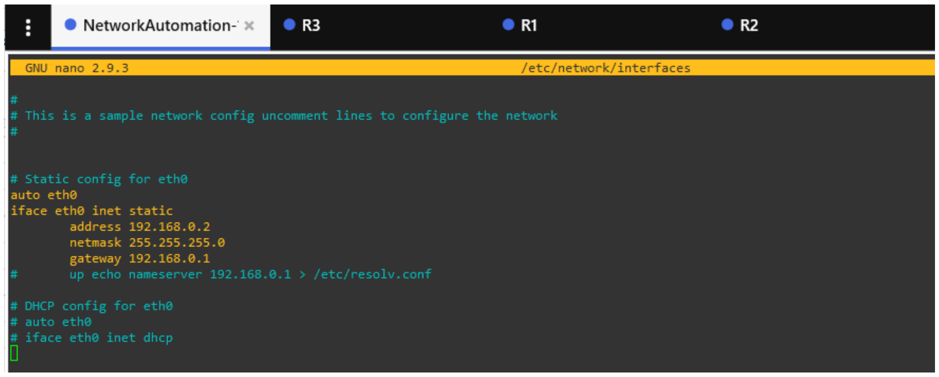


Figure 1: Network Topology

Preparing YAML Files

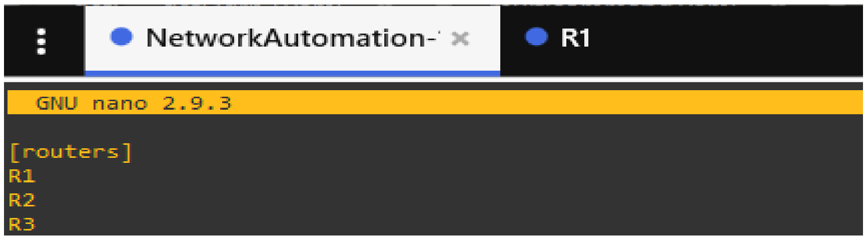
Some YAML files must be created before pushing the scripting on the devices: network interface configuration, ansible host and ansible configurations. The static IP address was configured inside *nano/etc/network/interfaces* files. Figure 2 shows the network interfaces configuration.



```
  GNU nano 2.9.3                                               /etc/network/interfaces

#
# This is a sample network config uncomment lines to configure the network
#


# Static config for eth0
auto eth0
iface eth0 inet static
        address 192.168.0.2
        netmask 255.255.255.0
        gateway 192.168.0.1
#       up echo nameserver 192.168.0.1 > /etc/resolv.conf

# DHCP config for eth0
# auto eth0
# iface eth0 inet dhcp
```

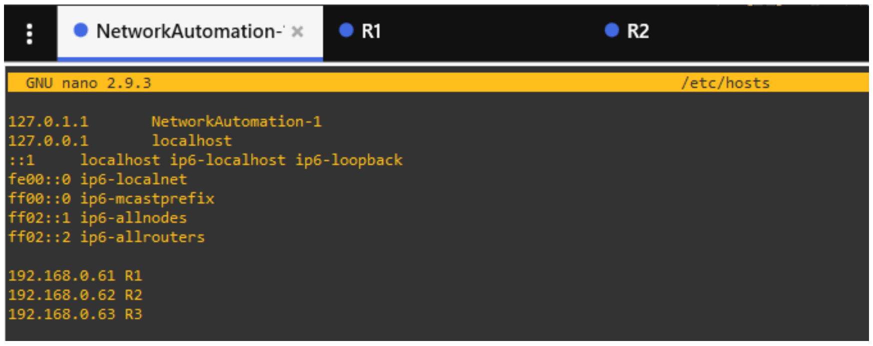
Figure 2: Network Interfaces Configuration

Ansible Host file configuration contains an inventory list that Ansible used to determine where the task should be performed and communicated through host name by adding the IP address belonging to the routers' interface. Figure 3 and Figure 4 show the inventory list and host file.



```
  GNU nano 2.9.3

[routers]
R1
R2
R3
```

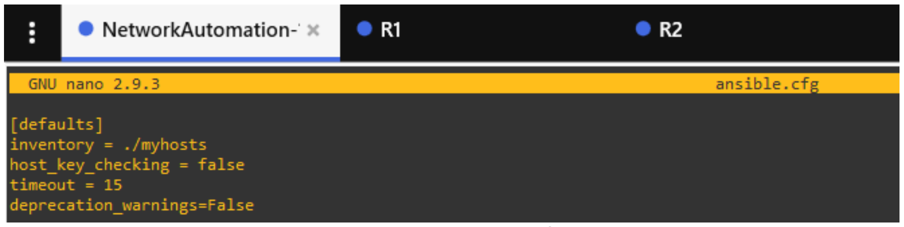
Figure 3: Inventory List



```
GNU nano 2.9.3                                    /etc/hosts

127.0.1.1       NetworkAutomation-1
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

192.168.0.61 R1
192.168.0.62 R2
192.168.0.63 R3
```

Figure 4: Host File

Lastly, Ansible.cfg file configuration was configured. A few lines in the Ansible configuration file specified which inventory to utilize. Figure 5 shows the Ansible.cfg.



```
GNU nano 2.9.3                                    ansible.cfg

[defaults]
inventory = ./myhosts
host_key_checking = false
timeout = 15
deprecation_warnings=False
```

Figure 5: Ansible.cfg

Ansible Playbook Scripting

The Ansible playbook is written in YAML format. The playbook is a YAML file that contains the command order. Tasks, modules, and files are all part of a role's structure. The role comprises a directory with subdirectories, each having a main.yml file that describes the order in which operations should be done. And the modules are short programs that perform specific activities on the system. They can be used alone or as part of larger scripts known as playbooks. Figure 6 shows the structure of Ansible Scripts.

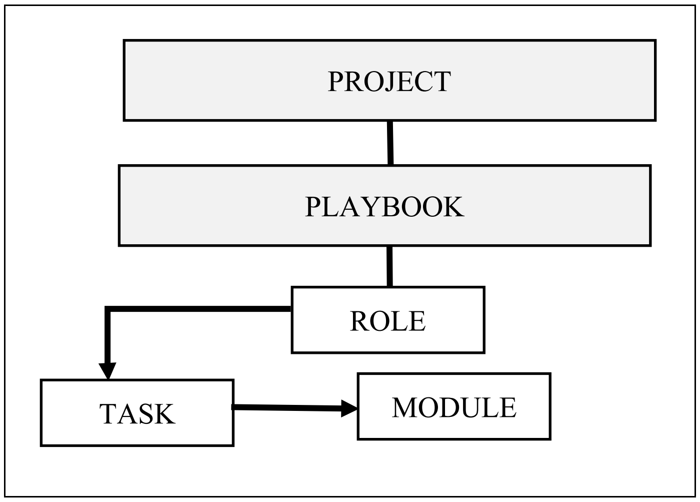


Figure 6: Structure of Ansible Scripts

Ansible scripting starts with configuration of the EIGRP routing protocol. The topology has three routers and is configured with routing to enable EIGRP. The Autonomous System (AS) ID 10 is configured on all directly connected interfaces. The next scripting is the configuration and propagated a default static route. Using the redistribute static command, a default static route is configured in R2 and propagated to all other routers. The last scripting is EIGRP fine-tune. The bandwidth, EIGRP interface percentage, hello interval, and hold timer are configured on all routers. The IP bandwidth-percent command is used to change the amount of bandwidth percentage available to EIGRP.

TESTING AND ANALYSIS

There are four tests involved in this project. For the first test, the Ansible playbook scripting was ran in network automation docker to check that all tasks such as IP address, loopback address, and basic EIGRP are working efficiently. For the second test, the EIGRP operation was conducted to verify the EIGRP configuration that consists of EIGRP neighbors, routing protocol information and routing table. The ping command and show run command were applied. The third test is the default static route test, which used show commands to view the task of propagating a default static route script on the router. The last test is Fine-Tune EIGRP test. This test was conducted to check the bandwidth utilization, hello interval and hold timer. Figure 7 shows the test framework used in this project.

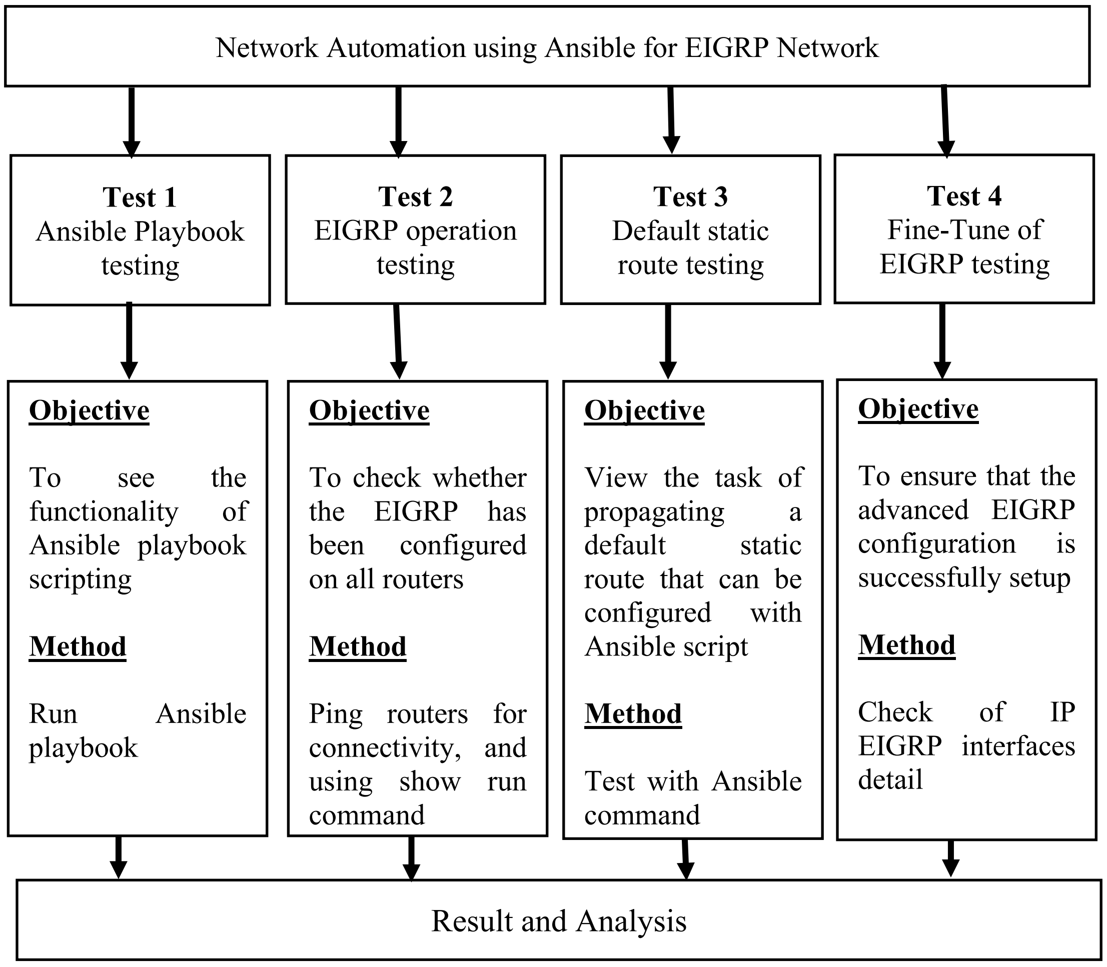
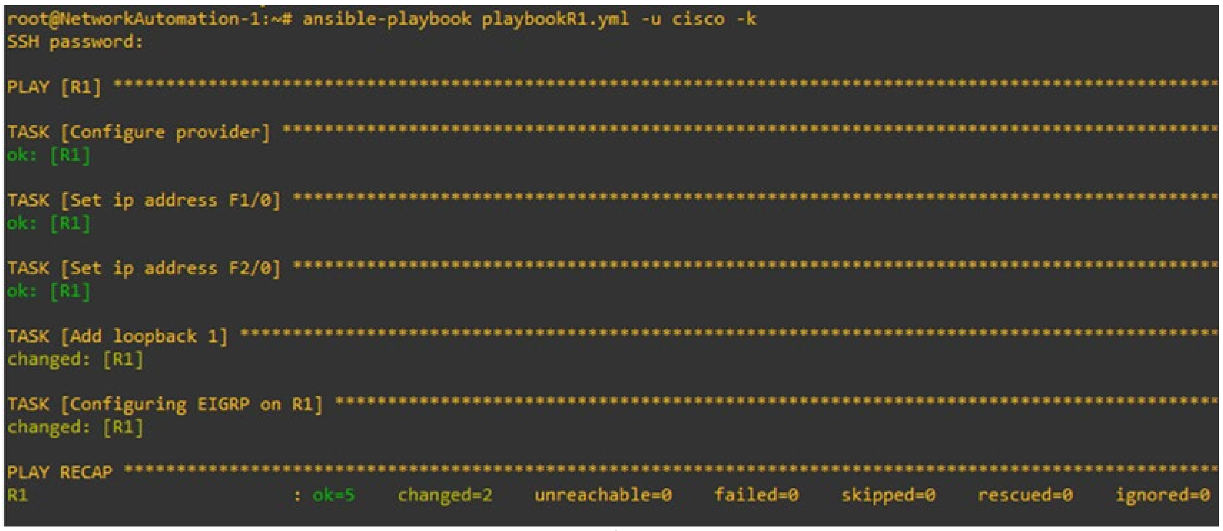

Network Automation using Ansible for EIGRP Network
Test 1
Ansible Playbook testing
Test 2
EIGRP operation testing
Test 3
Default static route testing
Test 4
Fine-Tune of EIGRP testing
Objective
To see the functionality of Ansible playbook scripting
Method
Run Ansible playbook
Objective
To check whether the EIGRP has been configured on all routers
Method
Ping routers for connectivity, and using show run command
Objective
View the task of propagating a default static route that can be configured with Ansible script
Method
Test with Ansible command
Objective
To ensure that the advanced EIGRP configuration is successfully setup
Method
Check of IP EIGRP interfaces detail
Result and Analysis

Figure 7: Test Framework

Ansible Playbook Results

The playbook contains one play with five tasks, and the output is shown in Figure 8. The result shows the four configurations in the target routers were successfully changed. The configurations included an interface IP address, a loopback IP address and an EIGRP configuration.



```
root@NetworkAutomation-1:~# ansible-playbook playbookR1.yml -u cisco -k
SSH password:

PLAY [R1] ***********************************************************************************

TASK [Configure provider] *******************************************************************
ok: [R1]

TASK [Set ip address F1/0] ******************************************************************
ok: [R1]

TASK [Set ip address F2/0] ******************************************************************
ok: [R1]

TASK [Add loopback 1] ***********************************************************************
changed: [R1]

TASK [Configuring EIGRP on R1] **************************************************************
changed: [R1]

PLAY RECAP **********************************************************************************
R1                         : ok=5    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 8: Result of Ansible Playbook

EIGRP Operation Results

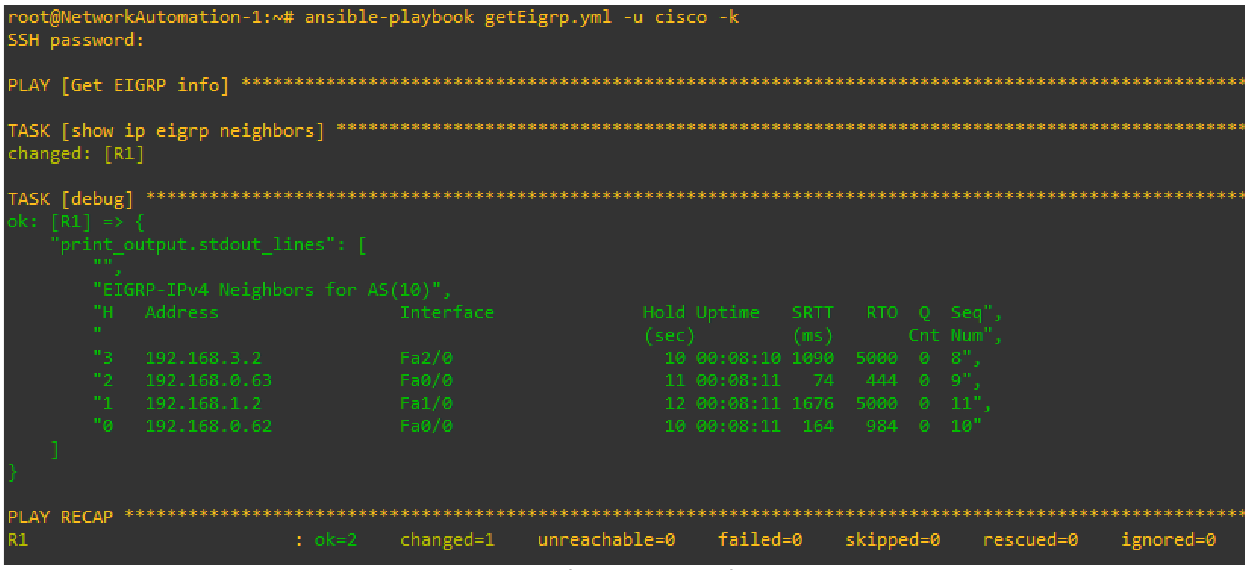
Based on Table 1 below, the routers could ping one another after EIGRP routing was configured. R1 could ping the R2 (192.168.0.62) and R3 (192.168.0.63). The ping results verified the successful connection.

Table 1: Ping Test Results

Router	Item	Description
R2	192.168.0.62	Success
R3	192.168.0.63	Success

Verification of EIGRP Neighbors

The result of each adjacent router's IP address and the interface used to reach the EIGRP neighbor can be verified with *show ip eigrp neighbors* command on R1. This command examines the neighbor table, confirms EIGRP is formed adjacent with R2 and R3 routers, and determines when neighbors become active and inactive. The result of the ansible-playbook *getEigrp.yml -u cisco -k* command also shows that the task played smoothly. Figure 9 shows the output of EIGRP neighbor scripting.



```
root@NetworkAutomation-1:~# ansible-playbook getEigrp.yml -u cisco -k
SSH password:

PLAY [Get EIGRP info] ***********************************************************************************

TASK [show ip eigrp neighbors] **************************************************************************
changed: [R1]

TASK [debug] ********************************************************************************************
ok: [R1] => {
    "print_output.stdout_lines": [
        "",
        "EIGRP-IPv4 Neighbors for AS(10)",
        "H   Address                 Interface              Hold Uptime   SRTT   RTO  Q  Seq",
        "                                                   (sec)         (ms)       Cnt Num",
        "3   192.168.3.2             Fa2/0                    10 00:08:10 1090  5000  0  8",
        "2   192.168.0.63            Fa0/0                    11 00:08:11   74   444  0  9",
        "1   192.168.1.2             Fa1/0                    12 00:08:11 1676  5000  0  11",
        "0   192.168.0.62            Fa0/0                    10 00:08:11  164   984  0  10"
    ]
}

PLAY RECAP **********************************************************************************************
R1                         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 9: EIGRP Neighbor Scripting

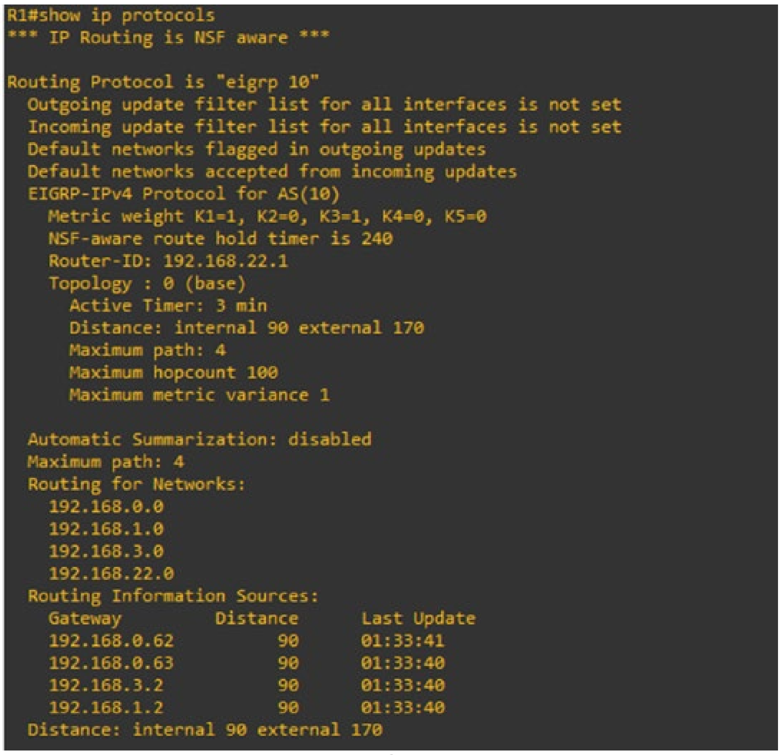
The comparison of both commands is shown in Table 2. It showed that all the configurations could be verified using *show ip eigrp neighbors* either by traditional technique or Ansible scripting.

Table 2: Testing Comparison

Type	Command	Task
Show run in router	show ip eigrp neighbors	Success
Ansible script	ansible-playbook getEigrp.yml -u cisco -k	Success

Routing Protocol Information

Show ip protocols command is used to display information about the routing protocol operation in R1. The output displayed the configuration, including the protocol, process ID and network. Figure 10 shows the IP address of the adjacent neighbors.



```
R1#show ip protocols
*** IP Routing is NSF aware ***

Routing Protocol is "eigrp 10"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP-IPv4 Protocol for AS(10)
    Metric weight K1=1, K2=0, K3=1, K4=0, K5=0
    NSF-aware route hold timer is 240
    Router-ID: 192.168.22.1
    Topology : 0 (base)
      Active Timer: 3 min
      Distance: internal 90 external 170
      Maximum path: 4
      Maximum hopcount 100
      Maximum metric variance 1

  Automatic Summarization: disabled
  Maximum path: 4
  Routing for Networks:
    192.168.0.0
    192.168.1.0
    192.168.3.0
    192.168.22.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.168.0.62          90      01:33:41
    192.168.0.63          90      01:33:40
    192.168.3.2           90      01:33:40
    192.168.1.2           90      01:33:40
  Distance: internal 90 external 170
```

Figure 10: EIGRP Protocols

Verification of EIGRP Routing Table

Verification of EIGRP routing table used *show ip route* and *ansible-playbook iproute.yml -u cisco -k* command. Table 3 shows the testing comparison result of both tests. It is proven that the configuration was successfully configured using Ansible playbook scripting.

Table 3: Testing Comparison

Type	Command	Task
Show run in router	show ip route	Success
Ansible script	ansible-playbook iproute.yml -u cisco -k	Success

Verification of Default Static Route

Default static route configuration was verified by using *show ip protocols* command on R2 and *ansible-playbook playbookR2.yml -u cisco -k* command in Ansible playbook. Table 4 shows the result for all routers using the *show ip route eigrp | include 0.0.0.0* command to view the default route's statement. The result represented the static default route. For R1 and R3, the D*EX indicated that they were external AS routes. The gateway of 0.0.0.0 means that there is no gateway for reaching the corresponding destination subnet. However, the administrative distance (AD) for all routers was 170.

Table 4: Default Route Statement

Type	Command	Administrative Distance
R1	Gateway of last resort is 192.168.1.2 to network 0.0.0.0 D*EX 0.0.0.0/0 [170/156160] via 192.168.1.2, 01:29:36, FastEthernet1/0	170
R2	Gateway of last resort is 0.0.0.0 to network 0.0.0.0	170
R3	Gateway of last resort is 192.168.2.1 to network 0.0.0.0 D*EX 0.0.0.0/0 [170/156160] via 192.168.2.1, 01:30:42, FastEthernet1/0	170

Verification of EIGRP Fine Tune

Table 5 shows the results obtained during the Fine Tune EIGRP testing. The advanced EIGRP testing for bandwidth utilization, hello interval, and hold time on R1 was tested with the ansible-playbook *FineTune.yml -u cisco -k* command. The *show ip eigrp interfaces detail* command was used to verify the configuration of *FineTune.yml* on all routers. The results show that the new bandwidth percentage changed to 75 percent and the hello-interval was 60 seconds, and the hold time was 180 seconds for the interfaces Fa1/0 and Fa2/0. EIGRP used no more than 75% of a link's available capacity. For every 60 seconds, the routers send out a hello packet to confirm its neighbor relationship, and if it does not receive a response, it will wait 180 seconds before announcing that neighbor drop.

Table 5: Parameters Details on R1

Interfaces	Hello-interval / seconds	Hold-timer / seconds	Bandwidth / percent
Fa0/0	5	15	-
Fa1/0	60	180	75
Fa2/0	60	180	-
Lo1	5	15	-

CONCLUSION AND RECOMMENDATIONS

The implementation of network automation using Ansible is simple and it is designed to assist network administrators in configuring the network devices. It helps the network administrator to manage devices efficiently and reduce the configuration time. The network administrator automated the scripts in the Ansible playbook and pushed to deploy the configurations or retrieve information from the managed devices. All the tasks that have been automated on the Ansible playbook are human-readable data format and agentless. The purpose of this project was to implement network automation using Ansible to configure EIGRP. Ansible was found to successfully automated the script and deployed the configurations. The results and analysis showed that the EIGRP configuration using automated scripts was verified and accurate. There are some recommendations for future research: implementing hardening and EIGRP authentication to improve security.

ACKNOWLEDGMENTS

The authors express sincere gratitude and thanks especially to the officers of IPD Kuala Muda for their contributions and supports in giving us information during interviews to complete our study and people who helps us direct and indirectly in this study.

CONFLICT OF INTERESTS DECLARATION

The authors declare no conflict of interests regarding the publication of this article.

REFERENCES

Biradar, A. G. (2020). A Comparative Study on Routing Protocols: RIP, OSPF and EIGRP and Their Analysis Using GNS-3. *Proceedings of the 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, 1-5.

Goyal, V. (2018). Review Paper on Comparison of RIP, OSPF, and EIGRP Protocols using Simulation. *International Journal on Future Revolution in Computer Science & Communication Engineering*, 4(4), 135–140.

Islami, M. F., Musa, P.,& Lamsani, M. (2020). Implementation of Network Automation using Ansible to Configure Routing Protocol in Cisco and Mikrotik Router with Raspberry PI. *Journal Ilmiah Komputer & Sistem Informasi (KOMPUTASI)*, 19(2), 127–134.

Manzoor, A., Hussain, M., &Mehrban, S. (2021). Performance Analysis and Route Optimization: Redistribution between EIGRP, OSPF & BGP Routing Protocols. *Journal of Computer Standards and Interfaces,* 68, 103391.

Masruroh, S. U., Robby, F., & Hakiem, N.(2016). Performance Evaluation of Routing Protocols RIPng, OSPFv3, and EIGRP in an IPv6 Network. *Proceedings of the International Conference on Informatics and Computing (ICIC)*, 111-116.

Mazin, A. M., Rahman, R. A., Kassim, M. & Mahmud, A. R. (2020). Performance Analysis on Network Automation Interaction with Network Devices using Python. *Proceedings of the 11th IEEE Symposium on Computer Application & Industrial Electronics (ISCAIE)*, 360-366.

Mihaila, P., Balan, T. C., Curpen, R., & Sandu, F. (2017). Network Automation and Abstraction using Python Programming Methods. *Proceedings of the 6th International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (MACRo),* 95–103.

Okonkwo, I. J., & Emmanuel, I. D. (2020). Comparative Study of EIGRP and OSPF Protocols based on Network Convergence. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 11(6), 39–45.

Ortiz-Garces, I., Echeverria, A. & Andrade, R. O. (2021). Automation Tasks Model for Improving Hardening Levels on Campus Networks. *Proceedings of the Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*, 30–35.

Shah, J., Dubaria, D., & Widhalm, J. (2018). A Survey of DevOps Tools for Networking. *Proceedings of the 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 185–188.

Wijaya, J. (2018). Network Automation using Ansible for Cisco Routers Basic Configuration. *Retrieved from osf.io/u8cdm.*