

## BAB 3

### METODE PENELITIAN

Metodologi Penelitian berisi uraian diagram alur penelitian. Diagram alur penelitian menjelaskan mengenai tahap-tahap penelitian. Dalam penelitian ini diperlukan alat pendukung untuk menunjang penelitian. Penelitian ini juga membutuhkan topologi yang berfungsi sebagai objek pengambilan data pada proses penelitian.

#### 3.1 Alat yang Digunakan

##### 3.1.1 Perangkat Keras (*Hardware*)

Penelitian ini akan menggunakan sebuah laptop sebagai perangkat keras, dengan spesifikasi yang dijelaskan pada tabel 3.1.

**Tabel 3.1 Spesifikasi Perangkat Keras**

|                                  |                        |
|----------------------------------|------------------------|
| OS                               | Windows 11             |
| <i>Processor</i>                 | Intel i3-6006U 2.0 GHz |
| <i>Random Acces Memory (RAM)</i> | 12 GB                  |
| <i>Storage (HDD)</i>             | 1 TB                   |
| <i>Storage (SSD)</i>             | 250 GB                 |

##### 3.1.2 Perangkat Lunak (*Software*)

*Tool* dan aplikasi perangkat lunak yang digunakan dalam penelitian ini dapat ditemukan pada tabel 3.2.

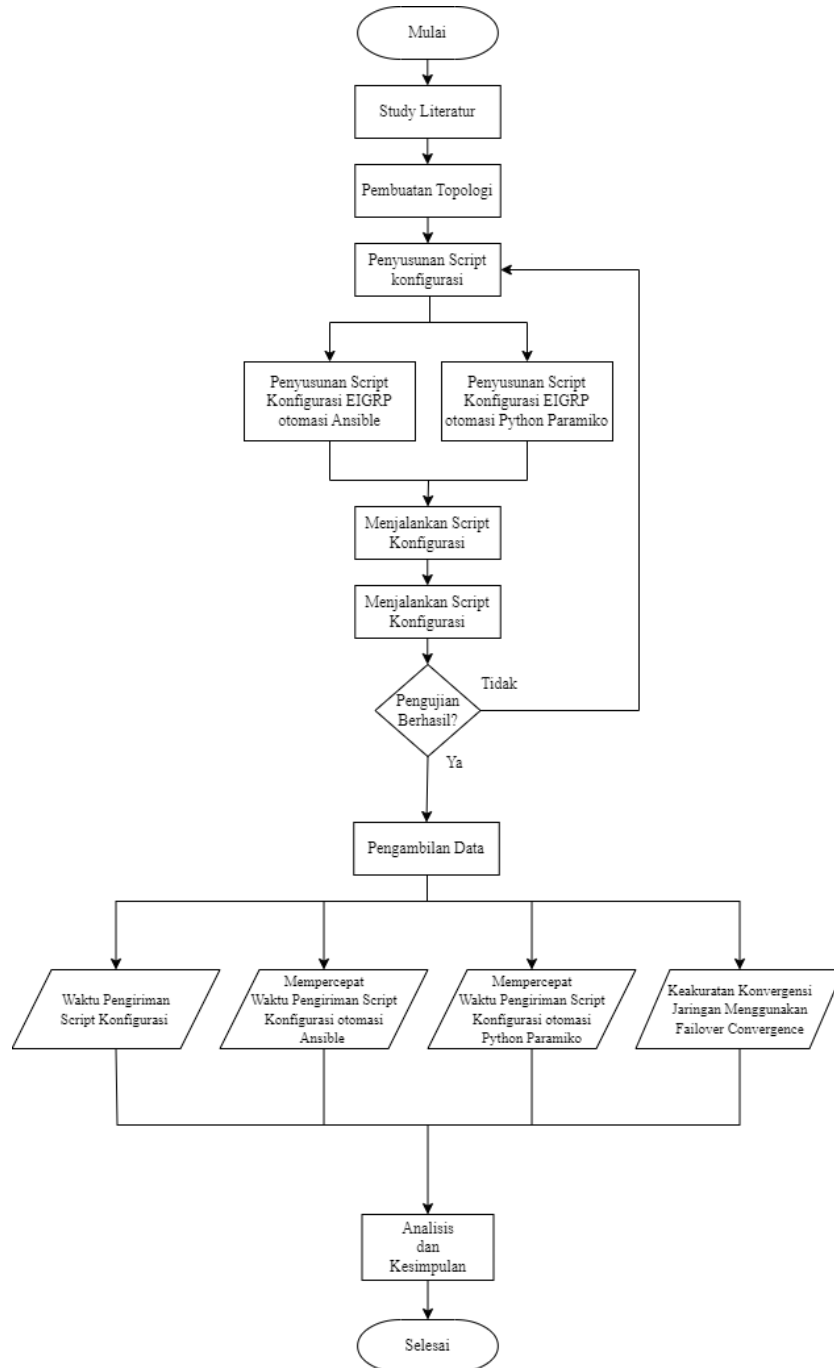
**Tabel 3.2 Perangkat Lunak**

| <i>Software</i>                     | Versi  | Fungsi                            |
|-------------------------------------|--------|-----------------------------------|
| GNS3                                | 2.2.38 | Penyusunan Topologi dan Pengujian |
| Vmware<br><i>Workstation</i><br>Pro | 17.0.1 | GNS3 VM                           |
| Wireshark                           | 4.0.3  | Pengambilan Data                  |

#### 3.2 Alur Penelitian

Tujuan dari penelitian ini adalah untuk melakukan simulasi *Network Automation* untuk mengkonfigurasi protokol *routing* EIGRP dengan

menggunakan otomasi Ansible dan otomasi Python paramiko melalui perangkat lunak GNS3. Penelitian dilakukan dengan mengikuti beberapa tahapan sesuai dengan alur diagram yang terdapat pada gambar 3.1.



**Gambar 3.1 Diagram Alur Penelitian**

Gambar 3.1 Menjelaskan alur perancangan sistem dalam penelitian ini. Tahapan pertama yang dilakukan yaitu studi literatur dari beberapa penelitian terkait dengan *Network Automation*, otomasi Ansible dan

otomasi Python Paramiko, *routing* protokol, serta penelitian lain yang terkait dengan penelitian ini. Dengan demikian tahapan ini berfungsi untuk memahami konsep dasar topik yang diambil.

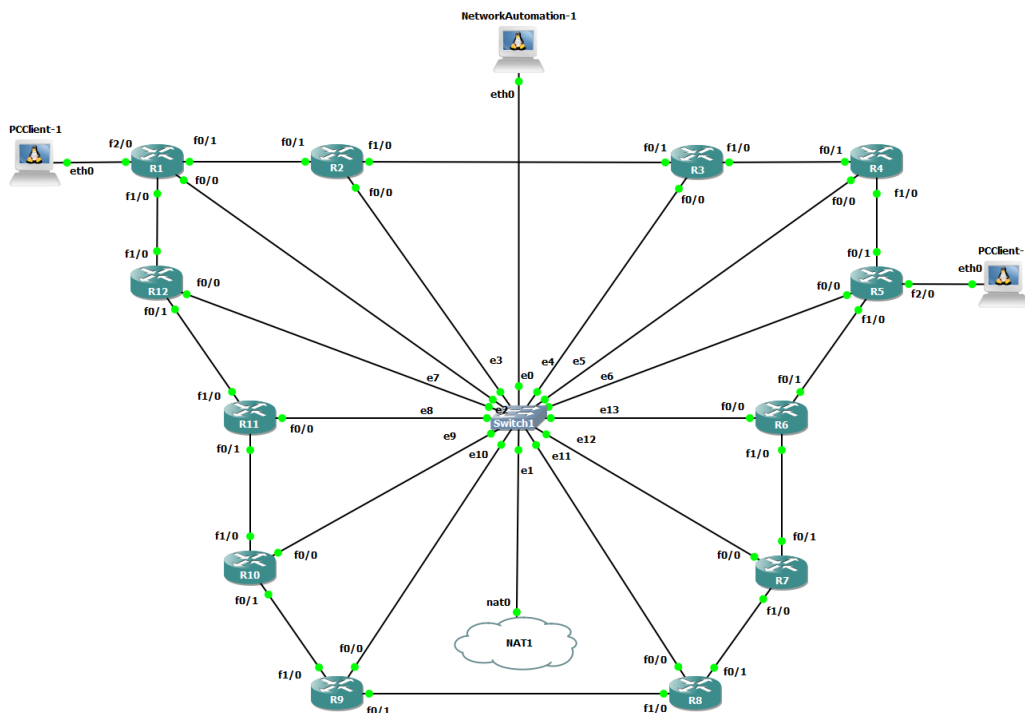
Langkah selanjutnya yaitu memasukan *Appliance Docker Network Automation* kedalam server GNS3 VM yang dilanjutkan pada proses pembuatan topologi jaringan pada GNS3 yang terdiri dari 1 *Docker Network Automation*, 1 buah *switch*, 12 buah Router, dan 2 buah PC Client. Setelah membuat topologi jaringan dilanjutkan dengan membuat menyusun *script* atau baris program menggunakan bahasa pemrograman YAML untuk otomasi Ansible dan Bahasa pemrograman Python untuk otomasi Python Paramiko, total *script* konfigurasi EIGRP yang dibuat berjumlah dua yaitu, *script* konfigurasi EIGRP otomasi Ansible untuk mengkonfigurasi protokol *routing* EIGRP dan *script* konfigurasi EIGRP otomasi Python Paramiko untuk mengkonfigurasi protokol *routing* EIGRP. Setelah program telah dibuat, langkah selanjutnya adalah melakukan eksekusi pada masing-masing program yang telah dibuat, untuk memeriksa apakah terdapat kesalahan atau tidak, perintah untuk mengeksekusi program yaitu “Ansible-*Playbook*” untuk program Ansible, dan perintah “python3” untuk program *library* paramiko. Jika terdapat *error* maka perlu kembali ke proses penyusunan program untuk memperbaiki *error* pada program yang telah dibuat, jika tidak terdapat *error* berarti pengujian berhasil.

Setelah berhasil menjalankan pengujian program tanpa kesalahan, langkah selanjutnya adalah mengambil data. Data yang diperlukan meliputi waktu yang dibutuhkan otomasi Python Paramiko dan otomasi Ansible untuk mengirimkan *script* konfigurasi EIGRP ke Router, metode untuk mempercepat otomasi Python Paramiko dan otomasi Ansible dalam mengirimkan *script* konfigurasi EIGRP ke Router dan keakuratan konvergensi jaringan protokol *routing* EIGRP menggunakan *failover convergence* setelah program diberikan oleh *Docker Network Automation* ke Router dalam jaringan. Proses pengambilan data ini dilakukan menggunakan perangkat lunak Wireshark dan GNS3. Setelah semua data terkumpul, langkah selanjutnya adalah melakukan analisis data. Analisis

data dilakukan dengan membandingkan grafik dari data waktu otomasi serta analisis konvergensi jaringan protokol routing EIGRP menggunakan *failover convergence*, dan kesimpulan diambil setelah proses analisis selesai dilakukan.

### 3.3 Rancangan Topologi

Dalam penelitian ini, digunakan topologi jaringan untuk mengotomatiskan jaringan pada *routing protocol* EIGRP. Topologi tersebut ditunjukkan pada gambar 3.2, di mana terdapat 12 Router Cisco 3725, satu *switch*, dan satu sistem *Docker Network Automation* yang membentuk susunan topologi tersebut. Fungsi dari *Docker Network Automation* adalah sebagai tempat untuk membuat dan mengirimkan *script* konfigurasi EIGRP menggunakan otomasi Ansible dan otomasi Python Paramiko ke 12 router melalui *switch*.



**Gambar 3.2 Topologi Jaringan EIGRP**

Gambar 3.2 menggambarkan topologi yang akan dipakai untuk menguji kinerja otomatisasi Ansible dan Python untuk *routing* EIGRP, dimana topologi yang digunakan merupakan topologi Mesh. Topologi tersebut tersusun dalam ASN 100 yang berisi R1, R2, R3, R4, R5, R6, R7,

R8, R9, R10, R11, R12. Semua router tersebut yang berjumlah 12 digunakan untuk menerima *script* konfigurasi EIGRP yang diberikan oleh *Docker Network Automation* menggunakan bantuan perangkat switch, yang dimana switch berfungsi sebagai pusat distribusi lalu lintas dari *Docker Network Automation* ke masing-masing router, setiap router akan terhubung ke switch melalui port yang telah ditentukan dan disesuaikan dengan konfigurasi, penggunaan NAT bertujuan untuk mendapatkan akses ke internet agar dapat menginstall otomasi Ansible dan otomasi Paramiko serta openssh server.

Melalui pengiriman *script* konfigurasi EIGRP tersebut, maka dapat diperoleh hasil kinerja dari otomasi Ansible dan otomasi Python Paramiko dan mengetahui cara atau metode untuk mempercepat otomasi Ansible dan otomasi Python Paramiko dalam pengiriman *script* konfigurasi EIGRP.

PC Client1 dan PC Client2 berfungsi untuk melakukan pengiriman data. Pengiriman data ini berupa paket ICMP yang berfungsi untuk mengetahui apakah PC Client1 dan PC Client2 terhubung melalui konfigurasi EIGRP serta untuk mengetahui konvergensi jaringan protokol routing EIGRP menggunakan *failover coonvergence*. Untuk melakukan konfigurasi memerlukan alamat IP yang sebelumnya sudah di masukkan di setiap PC pada jaringan yang digunakan agar saat melakukan konfigurasi EIGRP dapat saling terhubung. Alamat IP disetiap *Interface* yang terdiri dari Router, PC Client dan *Docker Network Automation* tertera pada Tabel 3.3.

**Tabel 3.3 Alamat IP Perangkat Jaringan**

| No | Device | Interface | IP Address       |
|----|--------|-----------|------------------|
| 1  | R1     | F0/0      | 192.168.0.11/24  |
|    |        | F0/1      | 10.10.10.1/30    |
|    |        | F1/0      | 120.120.120.2/30 |
|    |        | F2/0      | 192.168.1.2/24   |
| 2  | R2     | F0/0      | 192.168.0.12/24  |
|    |        | F0/1      | 10.10.10.2/30    |
|    |        | F1/0      | 20.20.20.1/30    |

| No | Device | Interface | IP Address       |
|----|--------|-----------|------------------|
| 3  | R3     | F0/0      | 192.168.0.13/24  |
|    |        | F0/1      | 20.20.20.2/30    |
|    |        | F1/0      | 30.30.30.1/30    |
| 4  | R4     | F0/0      | 192.168.0.14/24  |
|    |        | F0/1      | 30.30.30.2/30    |
|    |        | F1/0      | 40.40.40.1/30    |
| 5  | R5     | F0/0      | 192.168.0.15/24  |
|    |        | F0/1      | 40.40.40.2/30    |
|    |        | F1/0      | 50.50.50.1/30    |
|    |        | F2/0      | 172.16.1.2/24    |
| 6  | R6     | F0/0      | 192.168.0.16/24  |
|    |        | F0/1      | 50.50.50.2/30    |
|    |        | F1/0      | 60.60.60.1/30    |
| 7  | R7     | F0/0      | 192.168.0.17/24  |
|    |        | F0/1      | 60.60.60.2/30    |
|    |        | F1/0      | 70.70.70.1/30    |
| 8  | R8     | F0/0      | 192.168.0.18/24  |
|    |        | F0/1      | 70.70.70.2/30    |
|    |        | F1/0      | 80.80.80.1/30    |
| 9  | R9     | F0/0      | 192.168.0.19/24  |
|    |        | F0/1      | 80.80.80.2/30    |
|    |        | F1/0      | 90.90.90.1/30    |
| 10 | R10    | F0/0      | 192.168.0.20/24  |
|    |        | F0/1      | 90.90.90.2/30    |
|    |        | F1/0      | 100.100.100.1/30 |
| 11 | R11    | F0/0      | 192.168.0.21/24  |
|    |        | F0/1      | 100.100.100.2/30 |
|    |        | F1/0      | 110.110.110.1/30 |
| 12 | R12    | F0/0      | 192.168.0.22/24  |
|    |        | F0/1      | 110.110.110.2/30 |

|    |                           |      |                  |
|----|---------------------------|------|------------------|
|    |                           | F1/0 | 120.120.120.1/30 |
| 13 | <i>Network Automation</i> | Eth0 | 192.168.0.2/24   |
| 14 | PC Client1                | Eth0 | 192.168.1.1/24   |
| 15 | PC Client2                | Eth0 | 172.16.1.1/24    |

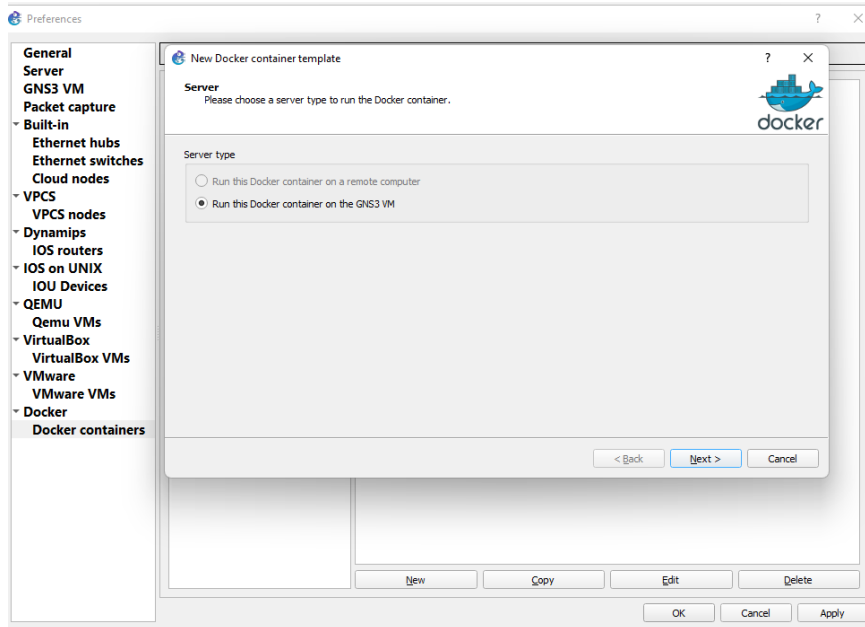
Tabel 3.3 memuat informasi mengenai identitas masing-masing perangkat seperti alamat IP dari Router, PC Client1, PC Client2, dan IP dari *Docker Network Automation* yang digunakan dalam jaringan.

### 3.4 Konfigurasi Perangkat

#### 3.4.1 *Docker Network Automation*

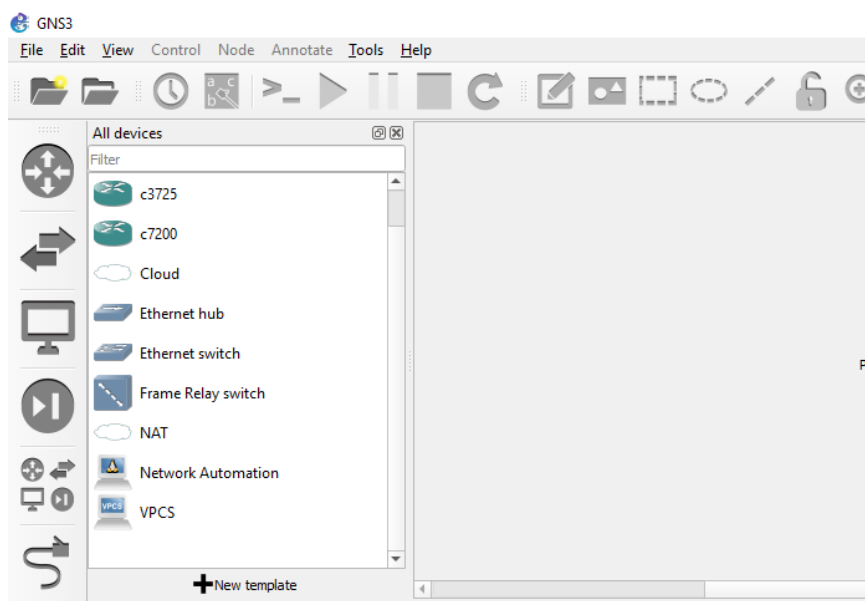
Pemasangan appliance *Docker Network Automation* merupakan langkah awal untuk membuat topologi yang bertujuan untuk mengotomasikan jaringan, *Docker Network Automation* digunakan sebagai tempat untuk menulis dan mengirim *script* konfigurasi EIGRP menggunakan otomasi Ansible dan otomasi Paramiko. Di dalam *Docker Network Automation* sudah terdapat otomasi jaringan seperti Ansible, Python Paramiko, Netmiko, atau alat lainnya untuk melakukan otomasi jaringan. sebelum melakukan pemasangan appliance peneliti mengunduh file appliance *Docker Network Automation* pada situs resmi GNS3

Pada gambar 3.3 merupakan langkah awal untuk pemasangan appliance *Docker Network Automation*, yang dimana terdapat pada menu “edit-preferences” pada aplikasi GNS3, selanjutnya pada bagian *Docker Container* tambahkan dengan cara klik “new”, lalu terdapat pilihan server yang akan menjadi server untuk *Docker Network Automation*, pada penelitian ini menggunakan server GNS3 VM.



**Gambar 3.3 Langkah Awal Pemasangan Appliance**

Langkah selanjutnya yang harus dilakukan yaitu memasukan file appliance yang sudah di unduh di situs resmi GNS3 maka gns3 akan melakukan proses pemasangan appliance. Jika proses pemasangan sudah selesai maka akan otomatis terlihat pada bagian devices aplikasi GNS3 ditunjukkan pada gambar 3.4.

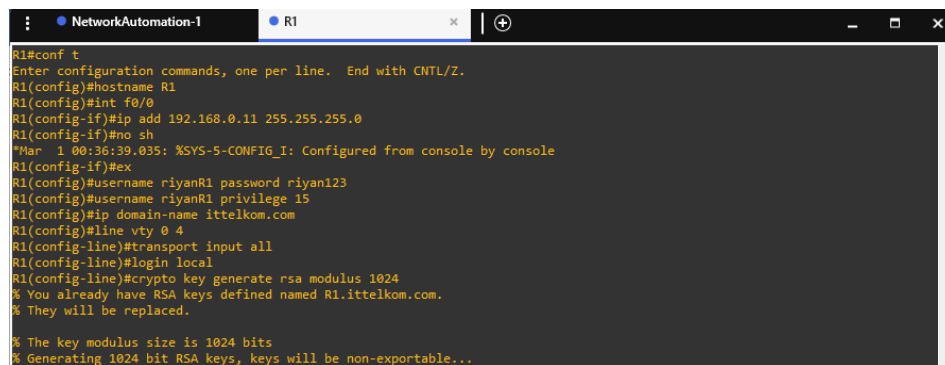


**Gambar 3.4 Pemasangan Appliance Sudah Berhasil**



### 3.4.2 Secure Shell (SSH)

Mengaktifkan protokol SSH pada masing-masing router, Konfigurasi SSH (*Secure Shell*) dilakukan untuk setiap Router yang ada pada topologi, supaya *Docker Network Automation* dapat berkomunikasi dengan semua Router. Dalam konfigurasi ini yang dilakukan adalah *Setting* alamat IP, pemberian *Username* dan *password*, *setting domain name*, *line vty* dan *setting Generate RSA Key* modulus untuk keperluan konfigurasi SSH pada *Interface Router* yang terhubung dengan *Docker Network Automation*



```
R1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#hostname R1
R1(config)#int f0/0
R1(config-if)#ip add 192.168.0.11 255.255.255.0
R1(config-if)#no sh
*Mar  1 00:36:39.035: %SYS-5-CONFIG_I: Configured from console by console
R1(config-if)#ex
R1(config)#username riyanR1 password riyan123
R1(config)#username riyanR1 privilege 15
R1(config)#ip domain-name ittelkom.com
R1(config)#line vty 0 4
R1(config-line)#transport input all
R1(config-line)#login local
R1(config-line)#crypto key generate rsa modulus 1024
% You already have RSA keys defined named R1.ittelkom.com.
% They will be replaced.
% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...
```

**Gambar 3.5 Konfigurasi Alamat IP Address dan SSH pada Router**

Konfigurasi pada gambar 3.5 berfungsi agar *Docker Network Automation* untuk dapat mengirimkan *script Ansible Playbook* dan *script library* paramiko ke setiap 12 Router yang terhubung. Pemberian konfigurasi alamat IP tersebut dilakukan ke setiap Router dengan alamat IP sesuai dengan tabel 3.3. Selain pemberian alamat IP dilakukan konfigurasi lain berupa pemberian *Username* dan *password*, *setting domain name*, *line vty* dan *setting Generate R SA Key* modulus untuk keperluan konfigurasi SSH, dilakukan agar *Docker Network Automation* dapat meremote atau mengakses setiap Router. Hal tersebut dilakukan agar *Docker Network Automation* dapat memberikan *script Ansible Playbook* dan *script library* paramiko kesetiap Router. Pemberian konfigurasi “IP domain-name ittelkom.com” digunakan untuk membuat DNS *domain name* pada Router. Untuk perintah “line vty 0 4” berfungsi untuk membatasi perangkat yang mengakses ssh tersebut (empat perangkat). Perintah “Transport input ssh” berfungsi sebagai perintah untuk menentukan perintah *protocol remote* berupa ssh. Perintah “login local” berfungsi untuk mengautentikasi

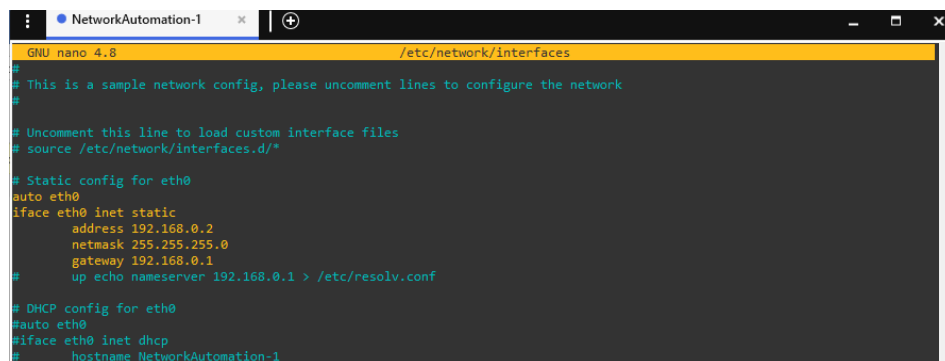
*Username* dan *password*. Dan yang terakhir Konfigurasi “*crypto Key generate rsa modulus 1024*” Pembuatan pasangan kunci RSA untuk Router dengan nilai ukuran modulus minimum 1024.

### 3.4.3 Otomasi Ansible

Pada otomasi Ansible langkah awal yang perlu diperhatikan yaitu beberapa *File* YAML harus dibuat sebelum mendorong *scripting* pada perangkat diantaranya : Konfigurasi *Network Interface*, konfigurasi Ansible *Host*, dan konfigurasi Ansible *File*.

#### A. *Network Interfaces*

Konfigurasi *Network Interfaces* merupakan langkah awal untuk menggunakan otomasi Ansible, yaitu mengkonfigurasi alamat ip untuk *Docker Network Automation* yang dimana akan digunakan untuk melakukan otomasi dengan otomasi Ansible, dengan cara menuliskannya pada file *interfaces* yang berada di direktori */etc/network*. Pada penelitian ini menggunakan alamat IP static ditunjukkan pada gambar 3.6.



```
GNU nano 4.8 /etc/network/interfaces
# This is a sample network config, please uncomment lines to configure the network
#
# Uncomment this line to load custom interface files
# source /etc/network/interfaces.d/*
# 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
#
#hostname NetworkAutomation-1
```

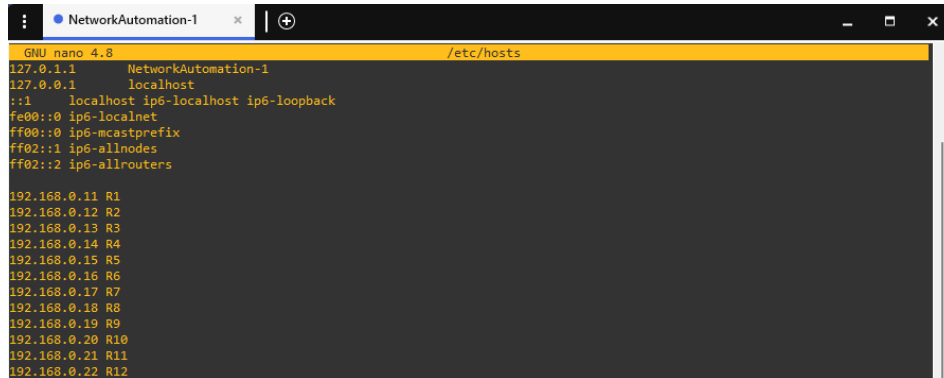
**Gambar 3.6** *File Network Interfaces*

Gambar 3.6 merupakan File Network Interfaces yang mendefinisikan konfigurasi IP statis untuk interface eth0. Dimana, interface eth0 akan dikonfigurasi dengan alamat IP statis 192.168.0.2, netmask 255.255.255.0, dan gateway 192.168.0.1.

#### B. *Hosts File Ansible*

*File Hosts* Ansible yang berada di directory */etc/* adalah *File* yang mencantumkan semua *Host* beserta alamat ip dari masing-masing *Host* yang ingin disambungkan, *file host* ini digunakan untuk *mapping* IP Address dan *device name* (fungsi DNS). Langkah awal yang dilakukan yaitu *setup* *Hostnya* dengan cara menambahkan *Host* beserta alamat identitas atau IP

kedalam *File Hosts* yang berada di direktori */etc/Hosts* guna melakukan konektifitas *Docker Network Automation* kepada Router ditunjukkan pada gambar 3.7.

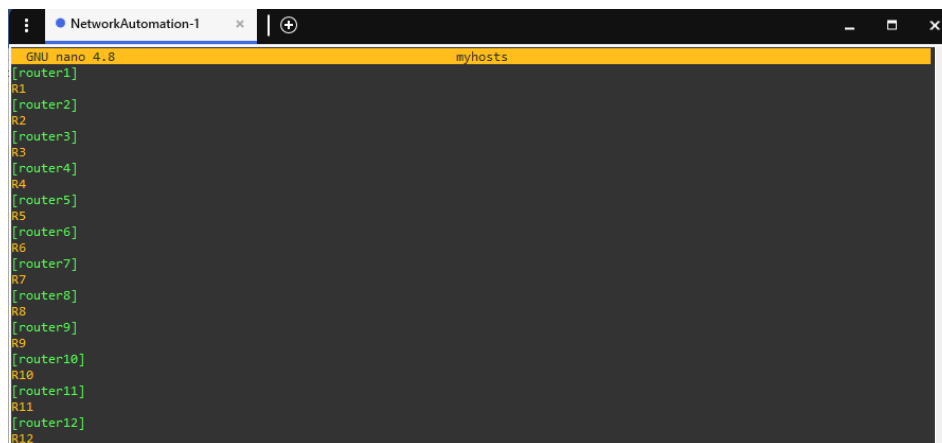


```
GNU nano 4.8 /etc/hosts
127.0.1.1 NetworkAutomation-1
127.0.0.1 localhost
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

192.168.0.11 R1
192.168.0.12 R2
192.168.0.13 R3
192.168.0.14 R4
192.168.0.15 R5
192.168.0.16 R6
192.168.0.17 R7
192.168.0.18 R8
192.168.0.19 R9
192.168.0.20 R10
192.168.0.21 R11
192.168.0.22 R12
```

**Gambar 3.7 File Host Directory etc**

Selanjutnya membuat *host file* di direktori *root* dengan nama *myhosts* yang akan digunakan untuk *inventory device* apa saja yang akan dimanage oleh Ansible atau *list inventory*, *Host* dapat dikelompokkan ke dalam grup seperti pada penelitian ini melakukan grouping untuk setiap *hostnya* agar lebih tertata dalam penulisan *script* konfigurasi EIGRP, yang dimana ditunjukkan pada gambar 3.8. *File myhosts* ini berbeda dengan *file host* di direktori *etc* pada gambar 3.7, yang digunakan untuk *mapping IP Address* dan *device name* (fungsi DNS).

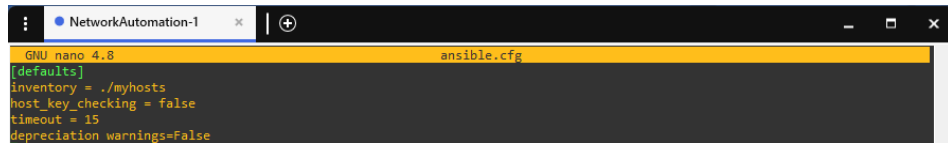


```
GNU nano 4.8 myhosts
[router1]
R1
[router2]
R2
[router3]
R3
[router4]
R4
[router5]
R5
[router6]
R6
[router7]
R7
[router8]
R8
[router9]
R9
[router10]
R10
[router11]
R11
[router12]
R12
```

**Gambar 3.8 File Host Directory root**

Langkah selanjutnya yaitu konfigurasi Ansible *File* di direktori *root* dengan nama *File Ansible.cfg* ditunjukkan pada gambar 3.9, Pada *File Ansible.cfg*, kita mendefinisikan bahwa *File inventory* adalah *myhosts*. *File inventory* ini adalah *File* yang menyimpan *list* dari *device* yang ingin kita

otomasi, *value* dari parameter *Host\_Key\_checking* bernilai *false*, saat *production* untuk alasan *security best practicenya* adalah diset *true*. *File* tempat *Host* yang digunakan adalah *File myhosts* di *root directory* → *./myhosts*, *Timeout ssh* = 5 menit.



```
GNU nano 4.8 ansible.cfg
[defaults]
inventory = ./myhosts
host_key_checking = false
timeout = 15
deprecation warnings=False
```

**Gambar 3.9 File Perintah Ansible Default**

### C. Skrip *Playbook* Ansible

Skrip program *Playbook* Ansible yang telah dibuat bertujuan untuk mengkonfigurasi alamat IP pada setiap *Interface* Router dan *routing* EIGRP. Program ini telah dimasukkan ke dalam lingkungan *Docker Network Automation* yang dapat diakses melalui perangkat lunak GNS3.

```
root@NetworkAutomation-1:~#nano eigrpansible.yml
```

Untuk membuat *File* *yml* di dalam *Docker Network Automation*, digunakan perintah seperti yang terlihat pada *script* diatas etelah itu, *File* program tersebut disimpan dengan nama "eigrpAnsible.yml" dan format yang sesuai.

```
---
- name: Konfigurasi EIGRP dan IP Address pada Router 1
  hosts: router1
  gather_facts: false
  connection: local
  vars:
    cli:
      username: riyanR1
      password: riyan123
      timeout: 100
  tasks:
    - name: Konfigurasi Interface fa0/1 ke Router 2
      ios_config:
        provider: "{{ cli }}"
```

parents: int fa0/1

lines:

- ip address 10.10.10.1 255.255.255.252
- no sh

register: print\_output

- debug: var=print\_output

- name: Konfigurasi Interface fa1/0 ke Router 12

ios\_config:

parents: int fa1/0

lines:

- ip address 120.120.120.2 255.255.255.252
- no sh

register: print\_output

- debug: var=print\_output

- name: Konfigurasi Interface fa2/0 ke PC Client1

ios\_config:

provider: "{{ cli }}"

parents: int fa2/0

lines:

- ip address 192.168.1.2 255.255.255.0
- no sh

register: print\_output

- debug: var=print\_output

```
- name: Konfigurasi EIGRP
  ios_config:
    provider: "{{ cli }}"
    parents: router eigrp 100
    lines:
      - network 10.10.10.0 0.0.0.3
      - network 120.120.120.0 0.0.0.3
      - network 192.168.1.0 0.0.0.255

  register: print_output

- debug: var=print_output
```

#### *Script Playbook Ansible untuk Konfigurasi Routing EIGRP*

*Script* diatas merupakan salah satu sample *script Playbook Ansible routing EIGRP*. Program tersebut dijalankan di topologi pada gambar 3.2. Pada *script "Hosts: Routers"* menunjukkan bahwa *Playbook* akan dijalankan pada grup *Host* bernama "Routers", "*gather\_facts: no*" menandakan bahwa Ansible tidak akan mengumpulkan informasi tentang *Host* sebelum menjalankan *Playbook*, "*ios\_config*" adalah modul Ansible yang digunakan untuk mengirimkan konfigurasi perangkat jaringan menggunakan protokol SSH, "*eigrp\_as*" adalah variabel yang digunakan untuk menyimpan nilai AS EIGRP yang akan digunakan, "*when: inventory\_hostname ==*" "R1" digunakan untuk membatasi eksekusi *task* hanya pada *Host* yang sesuai dengan nama yang ditentukan, misalnya hanya di R1, "*passive-Interface*" digunakan untuk menonaktifkan pengiriman paket *routing* melalui *Interface* tertentu, dan pada *script "backup: yes"* akan membuat *backup File* konfigurasi setelah *Playbook* selesai dijalankan.

#### **3.4.4 Mitogen Extension**

Pemasangan Mitogen extension bertujuan untuk mempercepat pengiriman *script* program konfigurasi otomasi ansible untuk digunakan sebagai metode dalam penelitian ini.

```

root@NetworkAutomation-1:
root@NetworkAutomation-1:
root@NetworkAutomation-1:
root@NetworkAutomation-1:
root@NetworkAutomation-1:
root@NetworkAutomation-1:~# curl -LO https://networkgenomics.com/try/mitogen-0.2.9.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  162  100  162    0     0   156      0  0:00:01  0:00:01 --:--:--  156
0    0    0    0    0     0    0      0  0:00:01  0:00:01 --:--:--    0
100 205k  100 205k    0     0  100k      0  0:00:02  0:00:02 --:--:--  450k
root@NetworkAutomation-1:~# ls
mitogen-0.2.9.tar.gz  myhosts
root@NetworkAutomation-1:~# tar -xvzf mitogen-0.2.9.tar.gz
mitogen-0.2.9/
mitogen-0.2.9/mitogen.egg-info/
mitogen-0.2.9/mitogen.egg-info/top_level.txt
mitogen-0.2.9/mitogen.egg-info/dependency_links.txt
mitogen-0.2.9/mitogen.egg-info/not-zip-safe
mitogen-0.2.9/mitogen.egg-info/SOURCES.txt
mitogen-0.2.9/mitogen.egg-info/PKG-INFO
mitogen-0.2.9/setup.cfg
mitogen-0.2.9/ansible_mitogen/
mitogen-0.2.9/ansible_mitogen/loaders.py
mitogen-0.2.9/ansible_mitogen/logging.py
mitogen-0.2.9/ansible_mitogen/plugins/
mitogen-0.2.9/ansible_mitogen/plugins/strategy/
mitogen-0.2.9/ansible_mitogen/plugins/strategy/mitogen_linear.py
mitogen-0.2.9/ansible_mitogen/plugins/strategy/mitogen.py

```

**Gambar 3.10** Instalasi Mitogen Extension

Gambar 3.10 merupakan langkah awal instalasi Mitogen extension, diawali dengan mengunduh file Mitogen di dalam *Docker Network Automation* menggunakan perintah "`curl -LO https://networkgenomics.com/try/mitogen-0.2.9.tar.gz`". Setelah selesai mengunduh, file Mitogen diekstrak dengan perintah "`tar -xvzf mitogen-0.2.9.tar.gz`" karena format *file* Mitogen adalah tar.gz.

```

GNU nano 4.8 ansible.cfg
[defaults]
inventory = ./myhosts
host_key_checking = false
timeout = 15
deprecation_warnings=False
strategy_plugins = ./mitogen-0.2.9/ansible_mitogen/plugins/strategy
strategy = mitogen_linear

```

**Gambar 3.11** File Perintah Ansible dengan Mitogen

Setelah selesai mengunduh dan meng-ekstract Mitogen extension, langkah selanjutnya ditunjukkan pada gambar 3.11, yaitu memasukan *script* untuk memanggil file Mitogen extension agar dapat digunakan untuk otomasi ansible dengan memasukan *script* "`strategy_plugins = ./mitogen-0.2.9/ansible_mitogen/plugins/strategy`" dan "`strategy = mitogen_linear`" dengan fungsi untuk menggantikan strategi eksekusi bawaan otomasi ansible Ansible atau default dengan strategi eksekusi yang diberikan oleh Mitogen yaitu linear atau berurutan, dengan cara memanggil file direktori Mitogen

extension. Dengan menggunakan extension Mitogen untuk mengotomasikan routing protokol EIGRP dengan otomasi Ansible maka peneliti dapat mempercepat pengiriman *script* dibandingkan dengan menggunakan bawaan atau default dari otomasi Ansible.

### 3.4.5 Otomasi Python Paramiko

Skrip program Python *library* paramiko yang telah dibuat bertujuan untuk mengkonfigurasi alamat IP pada setiap *Interface* Router dan *routing* EIGRP. Program ini telah dimasukkan ke dalam lingkungan *Docker Network Automation* yang dapat diakses melalui perangkat lunak GNS3.

```
root@NetworkAutomation-1:~#nano eigrpmiko.py
```

*Script* untuk membuat program otomasi Paramiko

Untuk membuat *File* paramiko di dalam *Docker Network Automation*, digunakan perintah seperti yang terlihat pada gambar 3.9. Setelah itu, *File* program tersebut disimpan dengan nama "eigrpmiko.py" dan format yang sesuai.

```
mport paramiko
import time

def R1():
    ip_address = '192.168.0.11'
    username = 'riyan'
    password = 'riyan123'

    ssh_client = paramiko.SSHClient()
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh_client.connect(hostname=ip_address, username=username,
password=password)

    print ("Login Sukses untuk Konfigurasi IP Address dan EIGRP pada Router 1
dengan IP {0}".format(ip_address))
    conn = ssh_client.invoke_shell()
```



```
conn.send ("conf t\n")
conn.send ("int fa0/1\n")
conn.send ("ip address 10.10.10.1 255.255.255.252\n")
conn.send ("no sh\n")
print("Konfigurasi Interface fa0/1 ke Router 2 !!SELESAI!!")
time.sleep (1)

conn.send ("conf t\n")
conn.send ("int fa1/0\n")
conn.send ("ip address 120.120.120.2 255.255.255.252\n")
conn.send ("no sh\n")
print("Konfigurasi Interface fa1/0 ke Router 12 !!SELESAI!!")
time.sleep (1)

conn.send ("conf t\n")
conn.send ("int fa2/0\n")
conn.send ("ip address 192.168.1.2 255.255.255.0\n")
conn.send ("no sh\n")
print("Konfigurasi Interface fa2/0 ke PC Client 1 !!SELESAI!!")
time.sleep (1)

conn.send ("router eigrp 100\n")
conn.send ("network 10.10.10.0 0.0.0.3\n")
conn.send ("network 120.120.120.0 0.0.0.3\n")
conn.send ("network 192.168.1.0 0.0.0.255\n")
print("Konfigurasi EIGRP pada Router 1 !!SELESAI!!")
time.sleep (1)

ssh_client.close()
```

R1 ()

*Script Program Python Paramiko untuk Konfigurasi Routing EIGRP*

*Script diatas* merupakan program konfigurasi *routing* EIGRP pada *Docker Network Automation* menggunakan *library* Paramiko. Pada *script* di atas, terdapat perintah *import* yang digunakan untuk mengimpor modul yang akan digunakan. Dalam program Paramiko, modul yang digunakan adalah modul "paramiko" itu sendiri, serta modul "time". Selanjutnya, terdapat deklarasi variabel "IP, Username, dan password". Variabel "Username" dan "password" berfungsi untuk menyimpan informasi Username dan password untuk akses SSH, sehingga Paramiko dapat mengakses perangkat yang akan di-remote menggunakan SSH. Perintah selanjutnya digunakan untuk memanggil fungsi SSH *client* dari Paramiko, serta menambahkan *Key policy* secara otomatis. *Command* "conn" berfungsi untuk memberikan perintah dari *script* Python ke Router yang diakses. "time.sleep(1)" merupakan perintah yang diberikan oleh *library* Paramiko untuk memberikan jeda 1 detik, agar menunggu Router selesai menjalankan perintah sebelum melanjutkan eksekusi perintah berikutnya.

### 3.4.6 Python Multi-Threading

Pemasangan modul Python Multi-Threading bertujuan untuk mempercepat pengiriman *script* program konfigurasi otomasi Python Paramiko untuk digunakan sebagai metode dalam penelitian ini.

```

root@NetworkAutomation-1:~#
root@NetworkAutomation-1:~# apt-get install -y python3-click-threading
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python3-click python3-colorama
The following NEW packages will be installed:
  python3-click python3-click-threading python3-colorama
0 upgraded, 3 newly installed, 0 to remove and 167 not upgraded.
Need to get 94.7 kB of archives.
After this operation, 424 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-colorama all 0.4.3-1build1 [23.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 python3-click all 7.0-3 [64.8 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/universe amd64 python3-click-threading all 0.4.4-2 [5992 B]
Fetched 94.7 kB in 2s (44.1 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package python3-colorama.
(Reading database ... 24957 files and directories currently installed.)
Preparing to unpack .../python3-colorama_0.4.3-1build1_all.deb ...
Unpacking python3-colorama (0.4.3-1build1) ...
Selecting previously unselected package python3-click.
Preparing to unpack .../python3-click_7.0-3_all.deb ...
Unpacking python3-click (7.0-3) ...
Selecting previously unselected package python3-click-threading.
Preparing to unpack .../python3-click-threading_0.4.4-2_all.deb ...
Unpacking python3-click-threading (0.4.4-2) ...
Setting up python3-colorama (0.4.3-1build1) ...
Setting up python3-click (7.0-3) ...
Setting up python3-click-threading (0.4.4-2) ...
root@NetworkAutomation-1:~#

```

**Gambar 3.12 Instalasi Python Multi-Threading**

. Langkah awal yang dilakukan ditunjukkan pada gambar 3.12, yaitu menginstall modul python multithreading dengan perintah “*apt-get install -y*

*python3-click-threading*” di dalam *Docker Network Automation*. Setelah modul sudah diinstal, selanjutnya yaitu memanggil modul *threading* ke dalam *script* program konfigurasi otomatis Python Paramiko, dengan menggunakan fungsi dari modul *threading* maka akan menggantikan eksekusi otomatis Python Paramiko bawaan atau default dengan eksekusi dari modul Python *Multi-threading* ditunjukkan pada contoh *script* berikut.

```
import paramiko
import time
import threading

def configure_router(ip_address, username, password, interfaces,
eigrp_networks):
    ssh_client = paramiko.SSHClient()
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh_client.connect(hostname=ip_address, username=username,
password=password)

    print("Login Sukses untuk Konfigurasi IP Address dan EIGRP pada Router
dengan IP {0}".format(ip_address))
    conn = ssh_client.invoke_shell()

    conn.send("conf t\n")

    for interface in interfaces:
        conn.send("int {0}\n".format(interface))
        conn.send("ip address {0} {1}\n".format(interfaces[interface]['ip'],
interfaces[interface]['netmask']))
        conn.send("no sh\n")
        print("Konfigurasi Interface {0} !!SELESAI!!".format(interface))
        time.sleep(3)

    conn.send("router eigrp 100\n")
```

```

for network in eigrp_networks:
    conn.send("network          {0}          {1}\n".format(network,
eigrp_networks[network]))
    print("Konfigurasi EIGRP pada Router !!SELESAI!!")
    time.sleep(3)

ssh_client.close()

# Membuat daftar berisi informasi untuk setiap router
routers = [
    {
        'ip_address': '192.168.0.11',
        'username': 'riyan',
        'password': 'riyan123',
        'interfaces': {
            'fa0/1': {'ip': '10.10.10.1', 'netmask': '255.255.255.252'},
            'fa1/0': {'ip': '120.120.120.2', 'netmask': '255.255.255.252'},
            'fa2/0': {'ip': '192.168.1.2', 'netmask': '255.255.255.0'},
        },
        'eigrp_networks': {
            '10.10.10.0': '0.0.0.3',
            '120.120.120.0': '0.0.0.3',
            '192.168.1.0': '0.0.0.255',
        }
    },
]

threads = []
for router_info in routers:
    t = threading.Thread(target=configure_router, args=(
        router_info['ip_address'],
        router_info['username'],
        router_info['password'],

```

```

router_info['interfaces'],
router_info['eigrp_networks'],
))
threads.append(t)

for t in threads:
    t.start()

for t in threads:
    t.join()

```

Penggunaan modul *threading* dalam Python bertujuan untuk mengonfigurasi router melalui koneksi SSH secara parallel, “*configure\_router1*” adalah fungsi untuk mengkonfigurasi Router 1 melalui koneksi SSH. Di dalam fungsi ini, terdapat perintah-perintah yang akan dieksekusi pada Router 1 untuk melakukan konfigurasi pada beberapa bagian seperti pengaturan alamat IP pada interface dan konfigurasi protokol routing EIGRP. Thread dengan nama “*thread1*” dibuat dengan menetapkan target fungsi “*configure\_router1*”. Thread dimulai dengan memanggil “*thread1.start()*”. Selanjutnya, “*thread1.join()*” digunakan untuk memastikan thread utama menunggu sampai thread *configure\_router1* selesai sebelum melanjutkan eksekusi. Dengan menggunakan modul Python Multi-Threading untuk mengotomasikan routing protokol EIGRP dengan otomasi Python Paramiko maka peneliti dapat mengirim *script* program konfigurasi secara parallel dan dapat mempercepat pengiriman *script* dibandingkan dengan menggunakan bawaan atau default dari otomasi Python Paramiko.

### 3.5 Percobaan *Script Network Automation*

Percobaan Program adalah langkah selanjutnya dalam mengotomatiskan jaringan. Dalam pengujian ini, Topologi yang digunakan yaitu mesh yang telah disusun menggunakan beberapa perangkat dan *Docker Network Automation*. Proses pengujian dilakukan dengan menjalankan *script* konfigurasi EIGRP menggunakan otomasi Ansible dan otomasi Python

Paramiko yang telah dibuat sebelumnya. *Script* tersebut berisi konfigurasi IP *Address* dan *routing protocol* EIGRP, yang diimplementasikan menggunakan otomasi Ansible dan otomasi Python Paramiko dalam format ".yml" dan ".py". Program tersebut akan dieksekusi pada topologi yang telah dibuat di aplikasi GNS3, dengan menggunakan solar-putty sebagai akses *remote* untuk menjalankan perintah.

```
root@NetworkAutomation-1:~# ansible-playbook eigrpansible.yml
```

Perintah untuk Menjalankan Program Konfigurasi Otomasi Ansible

```
root@NetworkAutomation-1:~# ansible-playbook eigrpansiblemitogen.yml
```

Perintah untuk Menjalankan Program Konfigurasi Otomasi Ansible dengan  
Mitogen Extension

```
root@NetworkAutomation-1:~#python3 paramikoeigrp.py
```

Perintah untuk Menjalankan Program Konfigurasi Otomasi Python Paramiko

```
root@NetworkAutomation-1:~#python3 paramikoeigrpmultithread.py
```

Perintah untuk Menjalankan Program Konfigurasi Otomasi Python Paramiko  
dengan Modul Multi-Threading

Empat skrip otomasi yang diuji dalam total program terdiri dari Program Konfigurasi Ansible untuk *routing* EIGRP, Program Konfigurasi Ansible metode Mitogen untuk *routing* EIGRP, Program Konfigurasi otomasi Python Paramiko untuk *routing* EIGRP, dan Program Konfigurasi otomasi Python Paramiko metode Multi-Threading untuk *routing* EIGRP. Program ini akan dijalankan dalam mode *root* pada *Docker Network Automation* dengan menggunakan perintah "*ansible-playbook*" untuk menjalankan Program konfigurasi otomasi Ansible, dan "*python3*" untuk menjalankan Program konfigurasi otomasi Python Paramiko. Program ini akan dijalankan dalam topologi EIGRP yang sesuai dengan gambar 3.2.

### 3.6 Skenario Pengujian dan Pengambilan Data

Langkah berikutnya adalah pengumpulan data, di mana data yang diperlukan dalam penelitian ini meliputi berapa waktu yang dibutuhkan otomasi otomasi Ansible dan Python Paramiko untuk mengirimkan *script* konfigurasi EIGRP ke Router, bagaimana mempercepat waktu waktu yang dibutuhkan otomasi Ansible dan otomasi Python Paramiko untuk

mengirimkan *script* konfigurasi EIGRP ke Router dan keakuratan konvergensi jaringan pada protokol *routing* EIGRP setelah program diberikan oleh *Docker Network Automation* ke Router dalam jaringan. Data akan diambil dari hasil pengujian program yang telah dijalankan, dan kemudian akan dibandingkan.

### 3.6.1 Waktu Pengiriman *Script* Konfigurasi EIGRP ke Router

Proses yang diperlukan dalam pengambilan waktu yang diperlukan oleh otomasi Ansible dan otomasi Python Paramiko saat memberikan *script* konfigurasi EIGRP ke setiap Router, dapat dilihat pada table 3.4.

**Tabel 3.4 Skenario Pengujian Waktu Pengiriman *Script* Konfigurasi EIGRP**

| Pengujian | <i>Network Automation</i> | Size <i>Script</i> | Banyak Pengujian |
|-----------|---------------------------|--------------------|------------------|
| 1         | Ansible                   | 11 kb              | 10 Kali          |
| 2         | Python (Paramiko)         | 14 kb              | 10 Kali          |

#### 1. Waktu Pengiriman *Script* Konfigurasi EIGRP ke Router Menggunakan Otomasi Ansible

Pada pengujian ini, waktu pengiriman *script* konfigurasi EIGRP menggunakan otomasi Ansible dianalisis menggunakan software Wireshark. *Script* konfigurasi EIGRP yang diotomatiskan menggunakan Ansible memiliki ukuran sebesar 11KB. Dalam pengujian ini, peneliti mencatat waktu awal akses SSH dan waktu akhir akses SSH saat program otomasi konfigurasi EIGRP dikirimkan dari *Docker Network Automation* menuju perangkat Router melalui perangkat switch. Dengan menggunakan waktu awal dan waktu akhir yang tercatat di Wireshark, peneliti dapat menghitung selisih waktu dengan mengurangi waktu akhir SSH dengan waktu awal akses SSH. Hal ini memberikan estimasi waktu yang diperlukan oleh otomasi Ansible untuk mencapai perangkat Router. Skenario pengujian dilakukan sebanyak sepuluh kali untuk mendapatkan data yang konsisten.

#### 2. Waktu Pengiriman *Script* Konfigurasi EIGRP ke Router Menggunakan Otomasi Python Paramiko

Pada pengujian ini, waktu pengiriman *script* konfigurasi EIGRP menggunakan otomasi Python Paramiko dianalisis menggunakan software

Wireshark. *Script* konfigurasi EIGRP yang diotomatiskan menggunakan otomasi Python Paramiko memiliki ukuran sebesar 14KB. Dalam pengujian ini, peneliti mencatat waktu awal akses SSH dan waktu akhir akses SSH saat program otomasi konfigurasi EIGRP dikirimkan dari *Docker Network Automation* menuju perangkat Router melalui perangkat switch. Dengan menggunakan waktu awal dan waktu akhir yang tercatat di Wireshark, peneliti dapat menghitung selisih waktu dengan mengurangi waktu akhir SSH dengan waktu awal akses SSH. Hal ini memberikan estimasi waktu yang diperlukan oleh otomasi Python Paramiko untuk mencapai perangkat Router. Skenario pengujian dilakukan sebanyak sepuluh kali untuk mendapatkan data yang konsisten.

Setelah peneliti memiliki waktu yang diperlukan oleh otomasi Ansible dan otomasi Python Paramiko, penulis membandingkan hasilnya, dan dapat mengidentifikasi otomasi yang lebih cepat antara keduanya berdasarkan perbandingan waktu yang diperlukan untuk menyelesaikan tugas.

### 3.6.2 Metode untuk Mempercepat otomasi Ansible dalam Mengirimkan *Script* Konfigurasi EIGRP ke Router

Proses yang diperlukan untuk pengambilan data pada metode untuk mempercepat otomasi Ansible dalam mengirimkan *Script* Konfigurasi EIGRP ke setiap Router terdapat dalam Tabel 3.5.

**Tabel 3.5 Skenario Pengujian Metode Mempercepat otomasi Ansible**

| Pengujian | <i>Network Automation</i> | <i>Size Script</i> | Metode  | Banyak Pengujian |
|-----------|---------------------------|--------------------|---------|------------------|
| 1         | Ansible                   | 11 kb              | Default | 10 Kali          |
| 2         | Ansible                   | 11 kb              | Mitogen | 10 Kali          |

Pada pengujian Metode untuk mempercepat waktu pengiriman *script* konfigurasi EIGRP pada otomasi Ansible, metode yang digunakan yaitu metode Mitogen, ekstensi Mitogen diinstal terlebih dahulu pada *Docker Network Automation* dengan perintah "*pip install mitogen*". Setelah Mitogen berhasil terpasang, langkah selanjutnya adalah memasukkan skrip konfigurasi Mitogen dengan cara mengedit file *ansible.cfg*. Konfigurasi



Mitogen dilakukan dengan menambahkan skrip "*strategy\_plugins = mitogen.strategy\_plugins*" agar Ansible menggunakan plugin Mitogen sebagai strategi eksekusi playbook. Selanjutnya, skrip Mitogen "*strategy = mitogen\_linear:*" digunakan untuk menentukan strategi Mitogen yang akan digunakan, yaitu "*mitogen\_linear*". Hal ini berarti tugas-tugas Ansible akan dieksekusi secara linear dengan menggunakan Mitogen guna meningkatkan kinerja dan efisiensi eksekusi. Setelah konfigurasi Mitogen selesai maka pengiriman *script* konfigurasi EIGRP dapat dilakukan.

Skenario pengujian dilakukan sebanyak sepuluh kali dengan menggunakan metode Mitogen, yang dimana dianalisis menggunakan software Wireshark. *Script* konfigurasi EIGRP yang diotomatiskan menggunakan metode Mitogen pada otomasi Ansible sama dengan otomasi Ansible default, yaitu sebesar 11KB. Peneliti mencatat waktu awal akses SSH dan waktu akhir akses SSH saat program otomasi konfigurasi EIGRP dikirimkan dari *Docker Network Automation* menuju perangkat Router melalui perangkat switch. Dengan menggunakan waktu awal dan waktu akhir yang tercatat di Wireshark, peneliti dapat menghitung selisih waktu dengan mengurangi waktu akhir SSH dengan waktu awal akses SSH. Hal ini memberikan estimasi waktu yang diperlukan oleh otomasi Ansible metode Mitogen untuk mencapai perangkat Router. Skenario pengujian dilakukan sebanyak sepuluh kali untuk mendapatkan data yang konsisten

Setelah peneliti memiliki waktu yang diperlukan oleh otomasi Ansible metode Mitogen dan otomasi Ansible default, penulis membandingkan hasilnya, dan dapat mengidentifikasi otomasi Ansible menggunakan metode Mitogen lebih cepat berdasarkan perbandingan waktu yang diperlukan untuk menyelesaikan tugas.

### **3.6.3 Metode untuk Mempercepat otomasi Python Paramiko dalam Mengirimkan *Script* Konfigurasi EIGRP ke Router**

Proses yang diperlukan untuk pengambilan data pada metode untuk mempercepat otomasi Python Paramiko dalam mengirimkan *Script* Konfigurasi EIGRP ke setiap Router terdapat dalam Tabel 3.6.

**Table 3.6 Skenario Pengujian Metode Mempercepat otomasi Python Paramiko**

| Pengujian | <i>Network Automation</i> | Size<br><i>Script</i> | Metode          | Banyak<br>Pengujian |
|-----------|---------------------------|-----------------------|-----------------|---------------------|
| 1         | Python (Paramiko)         | 14 kb                 | Default         | 10 Kali             |
| 2         | Python (Paramiko)         | 6,7 kb                | Multi-Threading | 10 Kali             |

Pada pengujian mempercepat waktu pengiriman *script* konfigurasi EIGRP pada otomasi Python Paramiko, metode yang digunakan yaitu metode Multi-Threading. Langkah awal yang dilakukan yaitu meng-import modul Multi-Threading ke dalam *script* program konfigurasi dengan *script* “*import threading*”, setelah modul di import, langkah selanjutnya yaitu membuat sebuah fungsi yang akan dieksekusi dalam thread terpisah bernama “*ssh\_thread*” dan berisi kode otomasi SSH menggunakan Paramiko, dengan *script* “*def ssh\_thread(hostname, username, password):*”, selanjutnya modifikasi *script* koneksi SSH agar dapat digunakan oleh Multithread dengan *script* “*{'hostname': 'host1', 'username': 'user1', 'password': 'pass1'}*”, selanjutnya masukan *script* “*t = threading.Thread(target=ssh\_thread, args=(host['hostname'], host['username'], host['password']))*” untuk memanggil *script* ssh yang sudah dimodifikasi, setelah menggunakan metode Multi-Threading selesai maka pengiriman *script* konfigurasi EIGRP dapat dilakukan. Dengan menggunakan metode Multi-Threading, maka otomasi Python Paramiko menjalankan beberapa koneksi SSH secara paralel, dan dapat meningkatkan efisiensi dalam melakukan pengiriman *script*.

Skenario pengujian dilakukan sebanyak sepuluh kali dengan menggunakan metode Multi-Threading, yang dimana dianalisis menggunakan software Wireshark. *Script* konfigurasi EIGRP otomasi Python Paramiko menggunakan metode Multi-Threading mempunyai ukuran *script* 6,7KB, yang dimana jauh lebih kecil dibandingkan dengan otomasi Python Paramiko default dengan ukuran *script* 14KB. Peneliti mencatat waktu awal akses SSH dan waktu akhir akses SSH saat program otomasi konfigurasi EIGRP dikirimkan dari *Docker Network Automation* menuju perangkat Router melalui perangkat switch. Dengan menggunakan waktu awal dan

waktu akhir yang tercatat di Wireshark, peneliti dapat menghitung selisih waktu dengan mengurangi waktu akhir SSH dengan waktu awal akses SSH. Hal ini memberikan estimasi waktu yang diperlukan oleh otomasi Python Paramiko metode Multi-Threading untuk mencapai perangkat Router. Skenario pengujian dilakukan sebanyak sepuluh kali untuk mendapatkan data yang konsisten.

Setelah peneliti memiliki waktu yang diperlukan oleh otomasi Python Paramiko metode Multi-Threading dan otomasi otomasi Python Paramiko default, penulis membandingkan hasilnya, dan dapat mengidentifikasi otomasi Python Paramiko menggunakan metode Multi-Threading lebih cepat berdasarkan perbandingan waktu yang diperlukan untuk menyelesaikan tugas.

### **3.6.4 Keakuratan Konvergensi Jaringan pada Routing Protokol EIGRP**

Proses yang diperlukan untuk pengambilan data keakuratan konvergensi jaringan pada routing protocol EIGRP yaitu menggunakan skenario *Failover Convergence* , dapat dilihat pada tabel 3.7, yang dimana terdapat tiga pengujian yaitu pengujian yang pertama Pencarian Jalur Terbaik (*Best Path*), Pengujian kedua Pencarian Jalur Cadangan (*Backup Path*) dan pengujian ketiga Waktu Kegagalan (*Failover*) pada Routing Protokol EIGRP menggunakan topologi pada gambar 3.2.

**Tabel 3.7 Skenario Pengujian Konvergensi Jaringan pada EIGRP**

| Pengujian | <i>Failover Convergence</i>                     | Banyak Pengujian |
|-----------|---|------------------|
| 1         | Pencarian Jalur Terbaik ( <i>Best Path</i> )    | 1 Kali           |
| 2         | Pencarian Jalur Cadangan ( <i>Backup Path</i> ) | 1 Kali           |
| 3         | Waktu Kegagalan ( <i>Failover</i> )             | 5 Kali           |

#### **I. Pencarian Jalur Terbaik (*Best Path*)**

Pada pencarian pencarian jalur terbaik (*best path*) analisis dilakukan menggunakan perintah “*Traceroute*” dari PC Client1 ke alamat IP address PC Client2, untuk melihat jalur mana yang dipilih sebagai jalur terbaik (*best path*), yang mana terdapat dua jalur pada topologi routing protokol EIGRP di dalam penelitian ini, yaitu jalur pertama dari R2 dengan alamat IP

10.10.10.2 dan jalur kedua dari R12 dengan alamat IP 120.120.120.1. Serta dilakukan pengecekan menggunakan perintah “*show ip route eigrp*” pada R1 untuk melihat nilai metric pada jalur yang dipilih, bertujuan untuk melihat apakah jalur terbaik (*best path*) yang dipilih sesuai dengan ketentuan routing protokol EIGRP, yaitu menggunakan algoritma DUAL menentukan jalur terbaik atau *successor* dengan perolehan nilai metric terendah.

## **2. Pencarian Jalur Cadangan (*Backup Path*)**

Pada pengujian pencarian jalur cadangan (*backup path*) analisis dilakukan menggunakan perintah “*Traceroute*” dari PC Client1 ke alamat IP address PC Client2 pada saat jalur terbaik (*best path*) dalam kondisi mati, untuk mendapatkan jalur cadangan (*backup path*), yang mana terdapat dua jalur pada topologi routing protokol EIGRP di dalam penelitian ini, yaitu jalur pertama dari R2 dengan alamat IP 10.10.10.2 dan jalur kedua dari R12 dengan alamat IP 120.120.120.1. Serta dilakukan pengecekan menggunakan perintah “*show ip route eigrp*” pada R1 untuk melihat nilai metric pada jalur yang didapat, bertujuan untuk melihat apakah jalur cadangan (*backup path*) yang didapatkan sesuai dengan ketentuan routing protokol EIGRP, yaitu menggunakan algoritma DUAL untuk menentukan jalur cadangan *feasible successor* dengan perolehan nilai metric tertinggi.

## **3. Waktu Kegagalan (*Failover*)**

Pada pengujian Keakuratan Konvergensi Jaringan EIGRP dengan *Failover Convergence*, analisis dilakukan dengan cara mematikan jalur yang sebelumnya sudah berjalan dengan baik atau jalur terbaik (*best path*), kemudian mengirimkan paket “ping” dari Client 1 ke Client 2 untuk mengukur waktu yang dibutuhkan oleh router pada protokol routing EIGRP untuk mengenali kegagalan dan beralih ke jalur cadangan (*backup path*). Skenario pengujian dilakukan sebanyak lima kali untuk mendapatkan data yang konsisten.